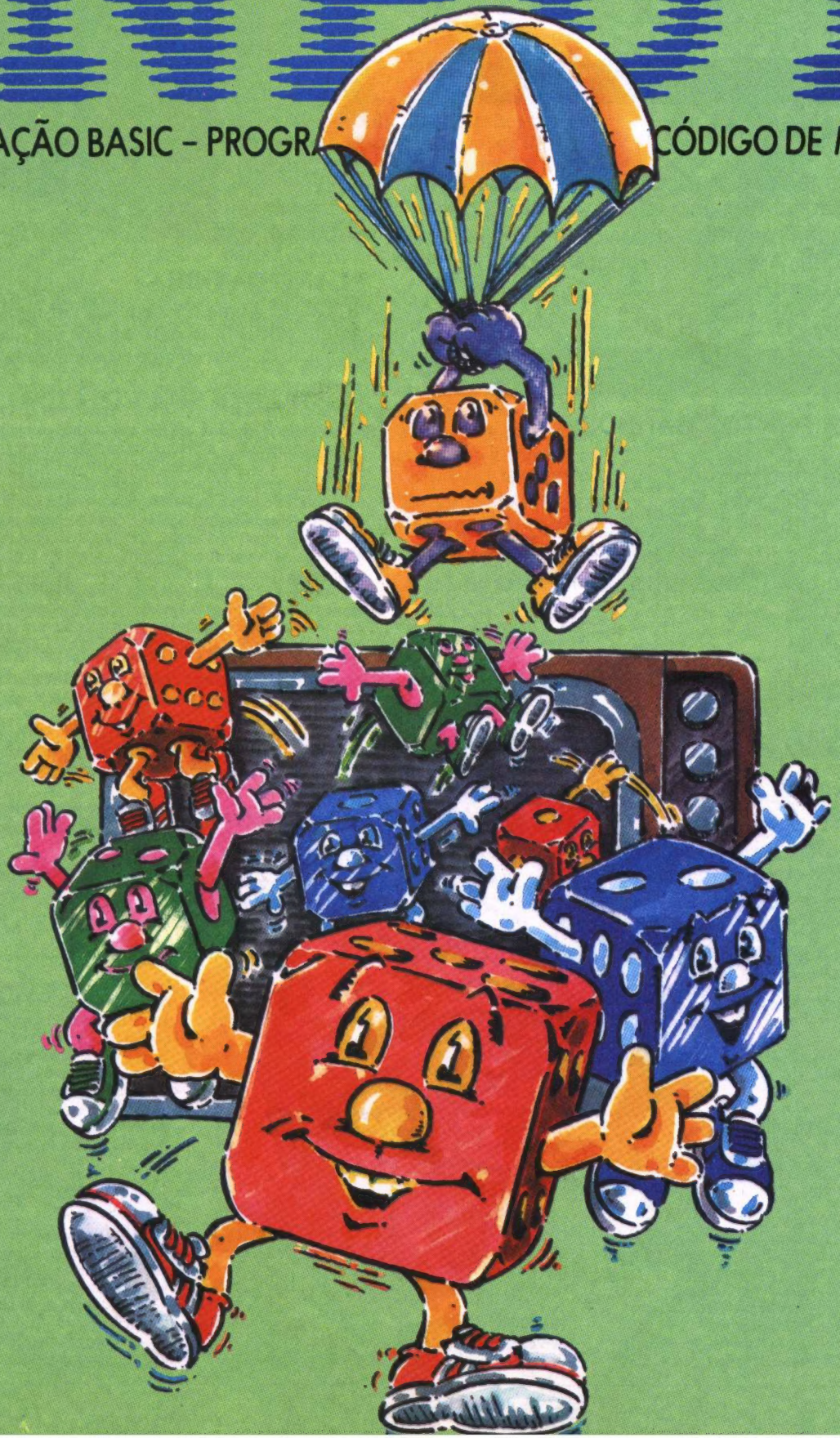
CURSO PRÁTICO 62 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC – PROGR CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 50,00



INPUT

Vol. 5 N.º 62

NESTE NÚMERO

PROGRAMAÇÃO BASIC

TÉCNICAS DE RECURSÃO

Recursão, uma técnica avançada de programação. Usos e dificuldades. Princípio básico: subdivisão do problema. Procedimentos recursivos. Previsão de erros. Limitações. Aplicações. Programação de jogos: *As Torres de Hanói*..... 1221

CÓDIGO DE MÁQUINA

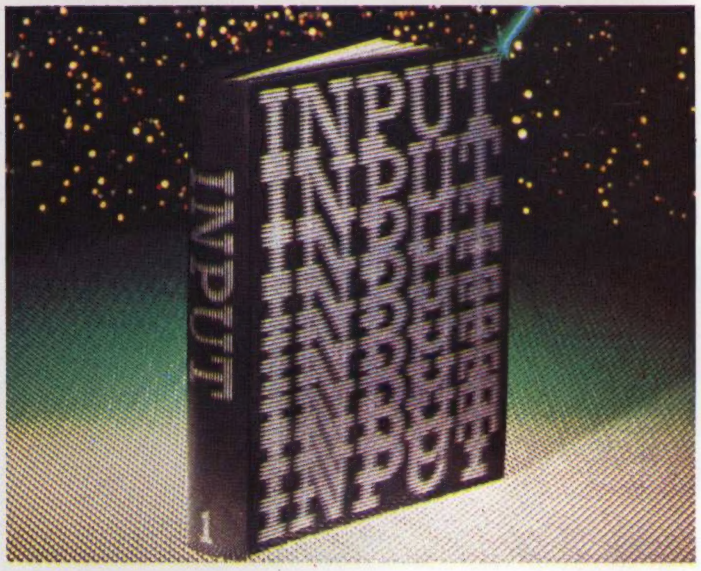
AVALANCHE: PONTOS GANHOS

Willie chega ao topo da montanha e recupera uma parte do lanche perdido. Execução da trilha sonora. Mudança do nível de dificuldade do jogo. Aumento da velocidade. Pontos ganhos. Incremento do placar. Alteração dos dígitos. Willie retorna ao ponto de partida 1228

PROGRAMAÇÃO DE JOGOS

OS DADOS VÃO ROLAR

Pôquer com dados: um jogo projetado para vários participantes. Regras. Estratégia. Inicialização. UDG dos dados. O laço-mestre. Rolando os dados. Cálculos do placar e verificação de entradas. Impressão do placar 1234



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; avenida Industrial, 117, Santo André; e no **Rio de Janeiro**: avenida Mem de Sá, 191/193, Centro. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida
Editora de Texto: Ana Lúcia B. de Lucena
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi
Secretária de Redação/ Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES

Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lucia de Britto
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

© Marshall Cavendish Limited 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

TÉCNICAS DE RECURSÃO

- UMA TÉCNICA MISTERIOSA?
- A MÁQUINA INTELIGENTE
- PROCEDIMENTOS RECURSIVOS
- LIMITAÇÕES
- COMO EVITAR ERROS

Se seu programa possui sub-rotinas que são chamadas repetidamente, é bem provável que as técnicas de recursão possam torná-lo mais rápido e eficiente. Veja como utilizá-las.

A construção de um programa consiste, essencialmente, em achar a melhor solução para um determinado problema através de uma série de comandos. À medida que ganha experiência, o programador aprende que o melhor caminho para isso é dividir o problema principal em partes menores. Tal método, no entanto, pode virar um pesadelo. Muitas vezes, ao tentar resolver uma etapa do processo, esbarra em outro processo e, quando resolve este, surge ainda outro e assim por diante. Em certo momento, o programador pensa em desistir, pois imagina estar andando em um verdadeiro círculo vicioso.

Mas um microcomputador não esmoreceria tão facilmente: para casos desse tipo, em que os problemas parecem sair uns dos outros, existem técnicas avançadas de programação, as chamadas *técnicas de recursão*.

UMA TÉCNICA MISTERIOSA?

Geralmente, programadores iniciantes consideram a recursão uma ferramenta complexa e até mesmo misteriosa. A grande dificuldade está na compreensão da própria listagem de um programa desse tipo; até mesmo os fluxogramas mais didáticos parecem obscuros quando representam processos recursivos. Apesar de tudo, o princípio básico dessa técnica não é tão complicado.

Na matemática, a palavra recursão é utilizada para designar a repetição de uma determinada operação. Em informática, porém, ela adquire um significado específico. Basicamente, a recursão é o chamamento sucessivo de uma sub-rotina ou de um procedimento, após serem estabelecidos alguns parâmetros iniciais. Depois de acionada, a sub-rotina — ou procedimento — chama a si mesma, repetidamente, atualizando os

parâmetros iniciais a cada vez, até que uma certa condição, previamente estabelecida, seja alcancada.

A MÁQUINA INTELIGENTE

Imaginemos a seguinte situação: um motorista se encontra em uma grande cidade desconhecida e precisa ir do lugar A para o lugar B. Embora tenha o mapa da cidade, acha muito difícil ir diretamente ao local desejado. Decide então dividir o itinerário em vários trechos, estudando um por vez. Escolhe, assim, um lugar C, em uma posição intermediária. Analisa o trajeto entre A e C e se dirige para lá.

Chegando em C, o motorista avalia a possibilidade de ir diretamente até B Caso isso lhe pareça complicado, procura uma outra posição intermediária, D. O processo se repete até que ele atinja seu destino.

Esse exemplo demonstra o princípio básico da recursão, tal como é aplicada em programação: a solução de um problema é obtida pela sua subdivisão em problemas menores ou mais fáceis.

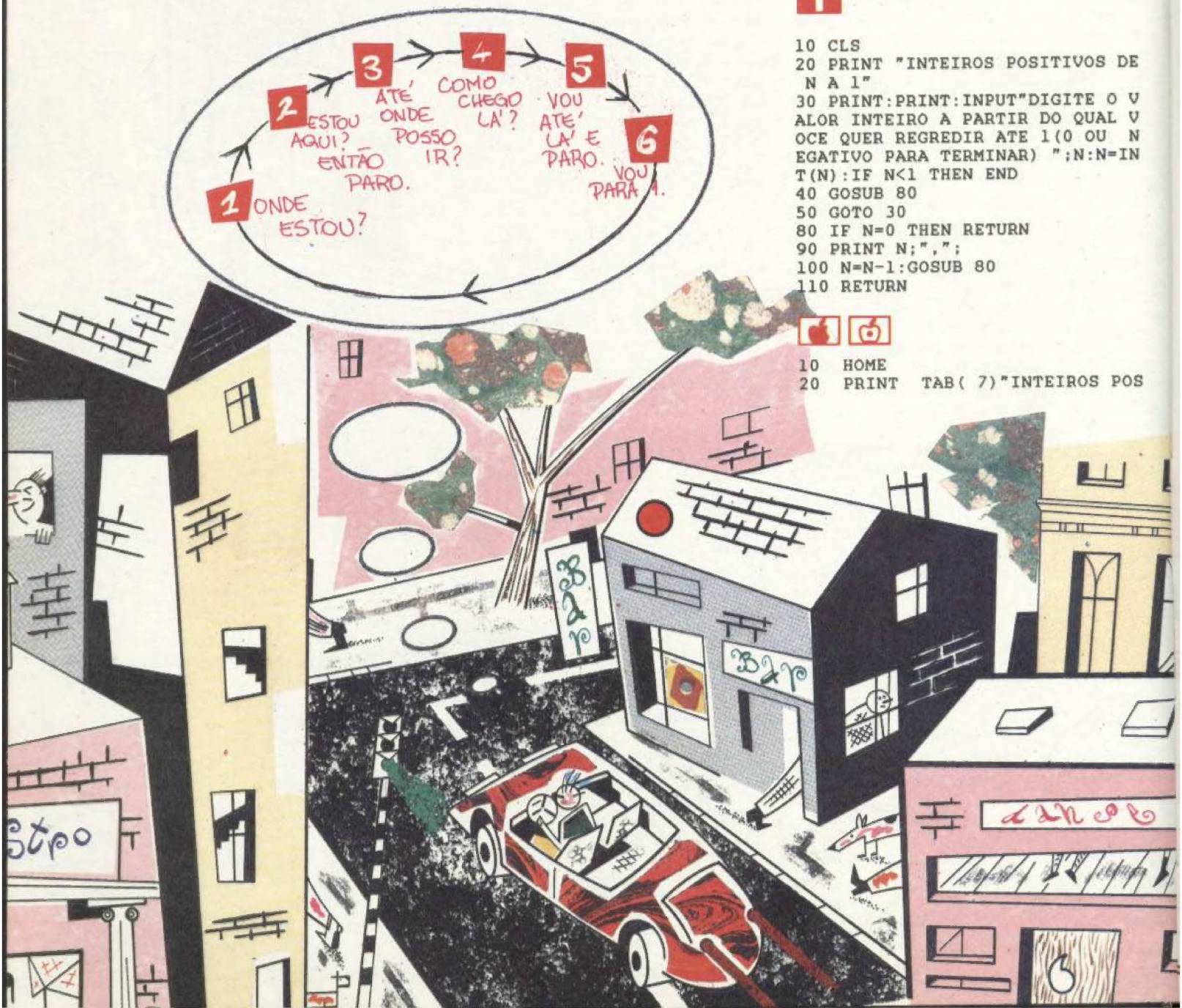
Assim que o programa alcança um novo nível de recursão, torna-se necessário armazenar as informações que foram conseguidas no nível anterior, para recuperá-las no passo seguinte. Ao início de cada nível, um novo conjunto de parâmetros é montado e, por meio de um teste, verifica-se se tal etapa chegou ao fim. Sem esse teste, o processo nunca terminaria.

Para observar o funcionamento dessa técnica, execute o próximo programa. Ele imprime todos os valores inteiros positivos de N até 1.

```
 20 PRINT  INVERSE 1;TAB 1;" I
NTEIROS POSITIVOS DE N ATE 1"
 30 INPUT "DIGITE O INTEIRO A
PARTIR DO    QUAL VOCE QUER C
ONTAR REGREDIN- DO ATE 1 (0 P
ARA SAIR) ";N: LET N=INT N:
IF N<1 THEN  STOP
 40 GOSUB 80
 50 GOTO 30
 80 IF N=0 THEN  RETURN
 90 PRINT ;N;"  ";
100 LET N=N-1: GOSUB 80
110 RETURN
```

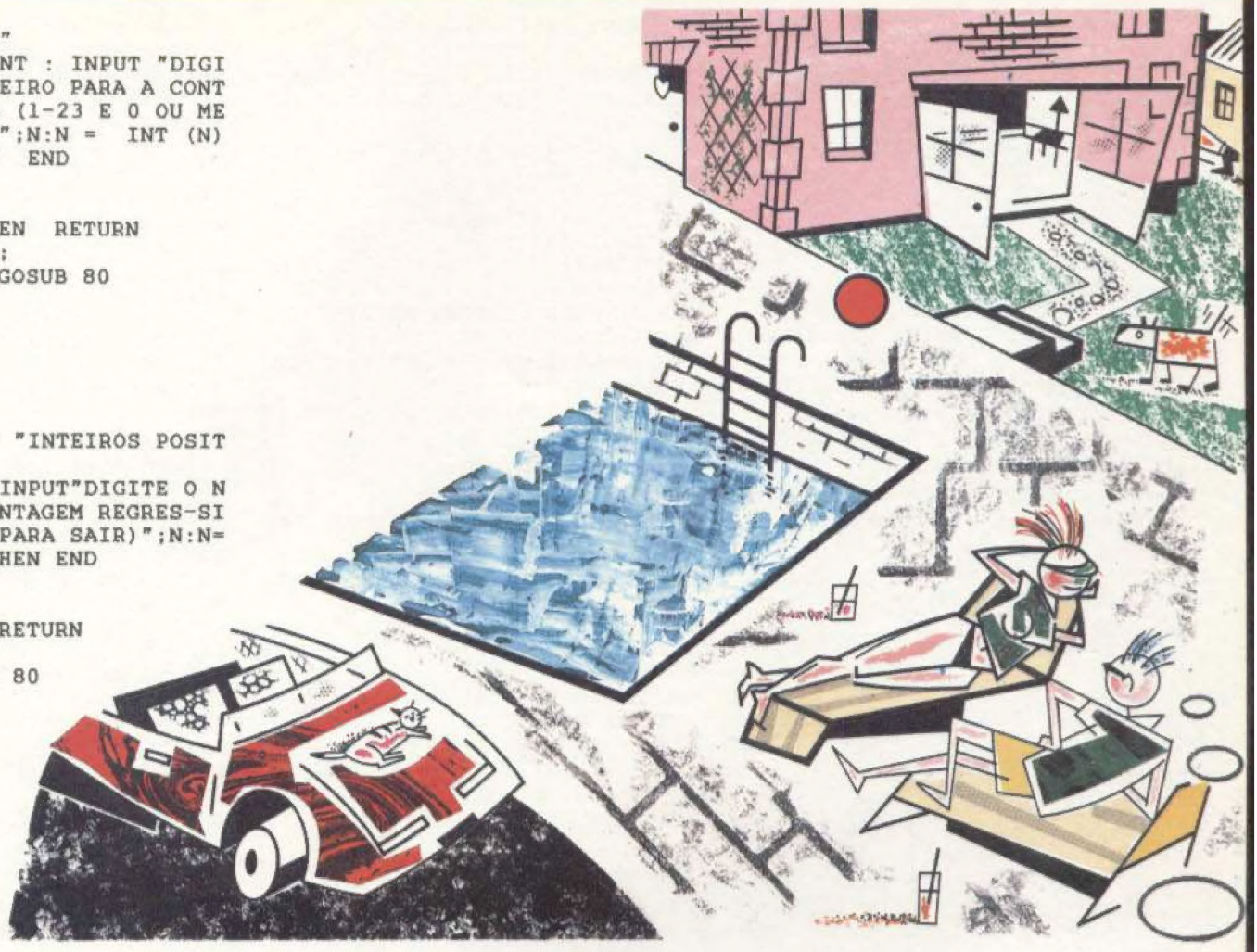
```
10 CLS
20 PRINT "INTEIROS POSITIVOS DE
 N A 1"
30 PRINT:PRINT:INPUT"DIGITE O V
ALOR INTEIRO A PARTIR DO QUAL V
OCE QUER REGREDIR ATE 1(0 OU  N
EGATIVO PARA TERMINAR) ";N:N=IN
T(N):IF N<1 THEN END
40 GOSUB 80
50 GOTO 30
80 IF N=0 THEN RETURN
90 PRINT N;",";
100 N=N-1:GOSUB 80
110 RETURN
```

```
10  HOME
20  PRINT  TAB( 7)"INTEIROS POS
```



```
ITIVOS DE 1 A N"
30  PRINT : PRINT : INPUT "DIGI
TE UM VALOR INTEIRO PARA A CONT
AGEM REGRESSIVA (1-23 E 0 OU ME
NOS PARA SAIR) ";N:N =  INT (N)
: IF N < 1 THEN  END
40  GOSUB 80
50  GOTO 30
80  IF N = 0 THEN  RETURN
90  PRINT N;",";
100 N = N - 1: GOSUB 80
110  RETURN
```

```
10 CLS
20 PRINT TAB(5) "INTEIROS POSIT
IVOS DE 1 A N"
30 PRINT:PRINT:INPUT"DIGITE O N
ÚMERO PARA A CONTAGEM REGRES-SI
VA (0 OU MENOS PARA SAIR)";N:N=
INT(N):IF N<1 THEN END
40 GOSUB 80
50 GOTO 30
80 IF N=0 THEN RETURN
90 PRINT N;",";
100 N=N-1:GOSUB 80
110 RETURN
```



O programa permite que você introduza um valor inteiro e, a partir dele, faz uma contagem regressiva até 1. Qualquer valor menor que 1 interrompe o programa. No Apple e no TK-2000, valores maiores que 23 não funcionarão, pois esses micros podem memorizar apenas 23 desvios para uma sub-rotina.

A linha 40 chama a sub-rotina recursiva. A primeira linha desta verifica se a tarefa foi completamente executada. Esse teste é essencial para o término de todo o processo.

No primeiro nível da recursão, a variável **N** possui o valor introduzido por você. Ele é impresso pela linha 90. O segundo nível é inicializado pela linha 100, que reduz o valor de **N** em uma unidade e chama a sub-rotina novamente, com o valor de **N** já alterado. O programa alterna-se, assim, repetidamente, entre as linhas 80 e 100.

Quando **N** alcança o valor 0 (na linha 100), o programa é desviado, como de costume, para a linha 80, onde encontra o comando **RETURN**, que o faz retornar da sub-rotina chamada na linha 100. A próxima instrução, na linha 110, também é um comando **RETURN**, que, desta vez, faz o programa retornar da sub-rotina chamada na linha 40. A linha 50 executa o programa novamente.

Note que quando o programa termina, a variável **N** contém o valor 0, atribuído na linha 100. Mas a linha 90 nunca imprime esse valor. Para colocar o último valor impresso na variável **N**, você poderia acrescentar a linha 105 com o comando **N=N+1** (**LET N=N+1**, para o Spectrum). Desse modo, **N** conteria sempre o último valor impresso.

PROCEDIMENTOS RECURSIVOS

Algumas outras linguagens, como o PASCAL, incluem o que se chama de *procedimento* (*procedure*). Trata-se de um conjunto de linhas definidas em uma certa parte do programa — como uma sub-rotina — que recebe um nome. Diferentemente das sub-rotinas, um procedimento é chamado através desse nome. Quando se define um procedimento, indicam-se também as variáveis que, após seu término, retornarão ao estado inicial. Este é um bom expediente na construção de processos recursivos.

Os tipos de BASIC com que trabalhamos não permitem que uma variável assuma um valor e, ao sair da sub-rotina, retorne ao valor inicial sem a interferência de um comando. Assim, somos obrigados a restabelecer seus valores iniciais antes de iniciar outra sub-rotina. O programa a seguir demonstra como contornar esse problema.

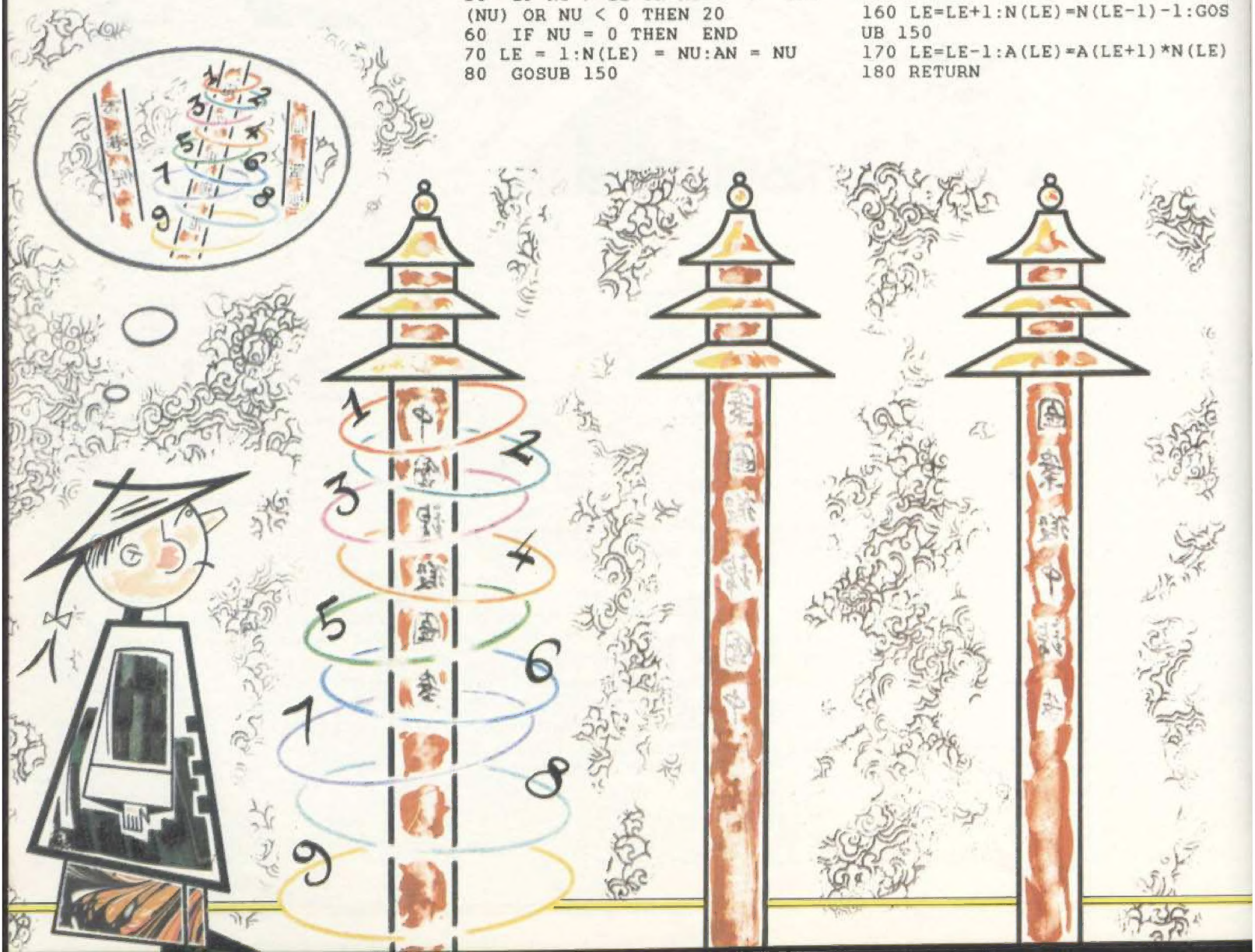
```
10 DIM N(34): DIM A(34)
20 CLS
```

```
30 PRINT TAB 6; INVERSE 1;"C
ALCULO DE FATORIAL"
40 INPUT "DIGITE NUMERO PARA
FATORIAL      (1-33, OU 0 PARA
 SAIR)";NU
50 IF NU>33 OR NU<>INT (NU)
OR NU<0 THEN  RUN
60 IF NU=0 THEN  STOP
70 LET LE=1: LET N(LE)=NU:
LET AN=NU
80 GOSUB 150
90 PRINT AN;"! = ";A(1):
PRINT : GOTO 40
150 IF N(LE)=0 THEN  LET A(LE)
=1: GOTO 180
160 LET LE=LE+1: LET N(LE)=N(L
E-1)-1: GOSUB 150
170 LET LE=LE-1: LET A(LE)=A(L
E+1)*N(LE)
180 RETURN
```

```
10 DIM N(34),A(34)
20 CLS
30 PRINT @6,"CALCULO DE FATORIA
L"
40 INPUT"DIGITE O NUMERO PARA F
ATORIAL   (1-33 OU 0 PARA SAIR)
   ";NU
50 IF NU>33 OR NU<>INT(NU) OR N
U<0 THEN 20
60 IF NU=0 THEN END
70 LE=1:N(LE)=NU:AN=NU
80 GOSUB 150
90 PRINT AN"! = ";A(1):PRINT:GO
TO 40
150 IF N(LE)=0 THEN A(LE)=1:GOT
O 180
160 LE=LE+1:N(LE)=N(LE-1)-1:GOS
UB 150
170 LE=LE-1:A(LE)=A(LE+1)*N(LE)
180 RETURN
```

```
10  DIM N(34),A(34)
20  HOME
30  PRINT  TAB( 10)"CALCULO DE
FATORIAIS"
40  PRINT : INPUT "DIGITE O NUM
ERO FATORIAL (1-22 OU 0 PARASAI
R) ";NU
50  IF NU > 22 OR NU <  >  INT
(NU) OR NU < 0 THEN 20
60  IF NU = 0 THEN  END
70 LE = 1:N(LE) = NU:AN = NU
80  GOSUB 150
90  PRINT AN"! = ";A(1): PRINT
: GOTO 40
150  IF N(LE) = 0 THEN A(LE) =
1: GOTO 180
160 LE = LE + 1:N(LE) = N(LE -
1) - 1: GOSUB 150
170 LE = LE - 1:A(LE) = A(LE +
1) * N(LE)
180  RETURN
```

```
10 DIM N(34),A(34)
20 CLS
30 PRINT TAB(9) "CALCULO DE FAT
ORIAIS"
40 PRINT:INPUT" DIGITE O NUMERO
 FATORIAL (1-33 E 0 OU MENOS PA
RA SAIR) ";NU
50 IF NU>33 OR NU<>INT(NU) OR N
U<0 THEN 20
60 IF NU=0 THEN END
70 LE=1:N(LE)=NU:AN=NU
80 GOSUB 150
90 PRINT:PRINT AN;"! = ";A(1):P
RINT:GOTO 40
150 IF N(LE)=0 THEN A(LE)=1:GOT
O 180
160 LE=LE+1:N(LE)=N(LE-1)-1:GOS
UB 150
170 LE=LE-1:A(LE)=A(LE+1)*N(LE)
180 RETURN
```



Execute o programa e introduza um valor qualquer. Lembre-se de que valores maiores que 22 não funcionarão no Apple, nem no TK-2000.

O programa calcula o fatorial do número que você digitou e o imprime na tela. O fatorial de um número é o resultado da multiplicação, entre si, de todos os números inteiros menores que ele e ele mesmo. Por exemplo, 5 fatorial (escreve-se 5!) é igual a 1 x 2 x 3 x 4 x 5, ou seja, 120. O cálculo de fatoriais geralmente é necessário em aplicações estatísticas. Um método rápido de executá-lo será sempre muito útil.

Inicialmente, o programa dimensiona algumas variáveis (linha 10), em número suficiente para gerar o fatorial máximo. A variável **N** terá como índice o nível da recursão **(LE) — N(LE)**.

A estrutura principal do programa começa na linha 70, onde o nível é inicializado em 1. Nessa mesma linha, o número que você introduziu — suponhamos que seja 5 — é atribuído a **N(1)** e a **AN** (variável que irá acumular a resposta). A linha 80 chama então o primeiro nível da recursão. A primeira linha da rotina de recursão (linha 150) verifica se a tarefa já chegou ao fim, ou seja, se **N(LE)** = 0. Como **N(LE)** ainda é 5, o programa passa para a linha 160, que incrementa o nível (para 2), estabelece como número corrente o 4 e chama a sub-rotina recursiva mais uma vez. Quando a linha 160 incrementar o valor do nível para 6 e decrementar o número corrente para 0, a linha 150 detectará esse último valor atribuindo 1 a **N(6)** e desviando o programa para a linha 180. Encontrando um **RETURN**, a execução passará para a linha 170, que foi a última a chamar a sub-rotina. Essa linha reduzirá o valor do nível para 5 e **A(5)** passará a ter o valor de **A(6)** vezes **A(5)**. Com isso, atribui-se a **A(5)** o valor de 1 x 1, ou seja, 1. A linha 180 devolve, então, o controle para o final da linha 150, onde **A(4)** assume o valor de **A(5)** vezes **A(4)**. Desse modo, será atribuído ao nível da recursão o valor 1 (calculado acima) vezes 2.

O laço irá se repetir sempre que o **GOSUB** da linha 160 for chamado. Quando o laço estiver completo, **LE** será 1 e o último **RETURN** apontará para a linha 80. A próxima instrução (linha 90) imprimirá então o resultado: 120.

LIMITAÇÕES

Além do problema com as variáveis, que não ocorre com os procedimentos, os tipos de BASIC com os quais trabalhamos apresentam uma limitação quanto ao número de vezes que uma sub-rotina pode chamar outra. A cada chamada o sistema precisa guardar na memória a posição de onde ela foi feita, o que requer a utilização de um ponteiro na pilha interna. É por esse motivo que o Apple e o TK-2000 não podem calcular fatoriais maiores que 22. Já o limite de 33 níveis para outros micros deve-se simplesmente à capacidade de armazenamento de números. O fatorial de 34 excede 1.7 x 10 ↑ 38, o máximo que um computador pode manipular.

PREVISÃO DE ERROS

Para manter seu programa dentro dos limites do micro e evitar uma "pane", você deve saber como ele se comporta desde o primeiro nível da recursão. Em geral, utiliza-se como teste uma informação inicial que culminará em uma resposta já esperada. Para facilitar o trabalho, você pode também considerar a rotina recursiva como um certo número de sub-rotinas em seqüência. Note que um desvio condicional para fora da sub-rotina é desaconselhável, pois o ponteiro que marca a volta do **GOSUB** ficaria desorientado.

Quando você estiver escrevendo uma sub-rotina recursiva, procure começar sempre com um teste de saída. A função desse teste é decidir se o problema (caracterizado pelos parâmetros iniciais) pode ser resolvido diretamente, sem a necessidade de subdivisões.

Antes de começar o programa, planeje com cuidado o que deseja que a sub-rotina faça. E, quando já estiver escrevendo o programa, não se preocupe com a seqüência exata da execução, levando em conta apenas dois princípios básicos: a condição de saída e a subdivisão em problemas menores.

Seguindo o método aqui apresentado, você será capaz de aplicar a recursão aos seus programas sem maiores dificuldades. Aqui está um exemplo de como essa técnica pode melhorá-los,tornando-os assim mais rápidos e compactos.

```
10 BORDER 1: INK 7: PAPER 1:
CLS
20 PRINT TAB 11; INVERSE 1;"
ORDENACAO "
30 INPUT "QUANTOS NUMEROS VOC
E QUER        ORDENAR (1-1000)
 ";A
40 IF A<1 OR A>1000 THEN
GOTO 10
50 DIM A(A): DIM R(2+SQR (A))
60 LET A(A)=100: PRINT
INVERSE 1;'"TABELA DESORDENAD
A :"'': FOR K=1 TO A-1: LET A
(K)=INT (RND*99): PRINT A(K);
" ";: NEXT K
70 LET L=1: LET LV=1: LET R=A
-1: GOSUB 1000
80 PRINT  INVERSE 1;'''"TABELA
 ORDENADA :"'': FOR K=1 TO A-
1: PRINT A(K);" ";: NEXT K
90 IF INKEY$<>" " THEN  GOTO
 90
100 RUN
1000 IF R>L THEN  LET I=L: LET
J=R+1: LET V=A(L): GOTO 1010
1005 RETURN
1010 LET I=I+1: IF A(I)<V THEN
 GOTO 1010
1020 LET J=J-1: IF A(J)>V THEN
 GOTO 1020
1030 IF J>=I THEN  LET T=A(I):
LET A(I)=A(J): LET A(J)=T: GOTO
 1010
1040 LET T=A(L): LET A(L)=A(J):
 LET A(J)=T
1050 LET R(LV)=R: LET LV=LV+1:
LET R=J-1: GOSUB 1000
1060 LET LV=LV-1: LET R=R(LV):
LET L=1: GOSUB 1000
1070 RETURN
```

```
10 CLS
20 PRINT @11,"ORDENACAO"
30 PRINT:INPUT" QUANTOS NUMEROS
 VOCE QUER        ORDENAR  (1-10
00)   ";A
40 IF A<1 OR A>1000 THEN 10
50 DIM A(A),R(1+SQR(A))
60 A(A)=100:PRINT" TABELA DESOR
DENADA :":FOR K=0 TO A-1:A(K)=R
ND(99):PRINT A(K);:NEXT:PRINT
70 L=0:R=A-1:GOSUB 1000
80 PRINT " TABELA ORDENADA :":F
OR K=0 TO A-1:PRINT A(K);:NEXT
90 IF INKEY$<>" " THEN 90:ELSE
RUN
1000 IF R>L THEN I=L:J=R+1:V=A(
L) ELSE RETURN
1010 I=I+1:IF A(I)<V THEN 1010
1020 J=J-1:IF A(J)>V THEN 1020
1030 IF J>=I THEN T=A(I):A(I)=A
(J):A(J)=T:GOTO 1010
1040 T=A(L):A(L)=A(J):A(J)=T
1050 R(LV)=R:LV=LV+1:R=J-1:GOSU
B 1000
1060 LV=LV-1:R=R(LV):L=I:GOSUB
1000
1070 RETURN
```

```
10  HOME
20  HTAB (11): INVERSE : PRINT
" ORDENANDO NUMEROS ": NORMAL
30  PRINT : INPUT " QUANTOS NUM
EROS A SEREM ORDENADOS ? (1-100
0) ";A
```

```
40  IF A < 1 OR A > 1000 THEN 1
0
50  DIM A(A),R(1 +  SQR (A))
60 A(A) = 100: PRINT : PRINT "
NUMEROS FORA DE ORDEM :-";: FOR
 K = 0 TO A - 1:A(K) =  INT (99
 *  RND (1)) + 1: PRINT A(K);"
";: NEXT : PRINT
70 L = 0:LV = 0:R = A - 1: GOSU
B 1000
80  PRINT " NUMEROS NA ORDEM :-
";: FOR K = 0 TO A - 1: PRINT A
(K);" ";: NEXT
90  GET I$: IF I$ <  > " " THEN
 90
100  RUN
1000  IF R > L THEN I = L:J = R
 + 1:V = A(L): GOTO 1010
1005  RETURN
1010 I = I + 1: IF A(I) < V THE
N 1010
1020 J = J - 1: IF A(J) > V THE
N 1020
1030  IF J >  = I THEN T = A(I)
:A(I) = A(J):A(J) = T: GOTO 101
0
1040 T = A(L):A(L) = A(J):A(J)
= T
1050 R(LV) = R:LV = LV + 1:R =
J - 1: GOSUB 1000
1060 LV = LV - 1:R = R(LV):L =
I: GOSUB 1000
1070  RETURN
```

```
10 CLS
20 PRINT TAB(10)"ORDENAÇÃO DE N
UMEROS"
30 PRINT:INPUT"QUANTOS NUMEROS
SERÃO ORDENADOS(1-1000)";A
40 IF A<1 OR A>1000 THEN 10
50 DIMA(A),R(1+SQR(A))
60 A(A)=100:PRINT"NUMEROS FORA
DE ORDEM:-":FOR K=0 TO A-1:A(K)
=INT(RND(-TIME)*99)+1:PRINT A(K
);:NEXT:PRINT
70 L=0:LV=0:R=A-1:GOSUB 1000
80 PRINT"NUMEROS ORDENADOS:":FO
R K=0 TO A-1:PRINT A(K);:NEXT
90 IF INKEY$<>" " THEN 90 ELSE
RUN
1000 IF R>L THEN I=L:J=R+1:V=A(
L):GOTO 1010
1005 RETURN
1010 I=I+1:IF A(I)<V THEN 1010
1020 J=J-1:IF A(J)>V THEN 1020
1030 IF J>=I THEN T=A(I):A(I)=A
(J):A(J)=T:GOTO 1010
1040 T=A(L):A(L)=A(J):A(J)=T
1050 R(LV)=R:LV=LV+1:R=J-1:GOSU
B 1000
1060 LV=LV-1:R=R(LV):L=I:GOSUB
1000
1070 RETURN
```

PROGRAMA DE ORDENAÇÃO

Compare a listagem do método de ordenação tipo bolha do artigo da página 468 com esta, que utiliza a recursão. Nosso programa não contém tantos desvios condicionais do tipo **IF...THEN, GOTO...**, nem tantas variáveis.

Inicialmente, o programa pede que você introduza a quantidade de números randômicos que serão ordenados. A linha 50 dimensiona a matriz **A(A)**, que armazena esses números, e a matriz **R**, que irá guardar o valor das variáveis durante o processo recursivo. A linha 60 gera e imprime os números randômicos, ainda fora de ordem, e a linha 70 chama a sub-rotina recursiva para ordenálos. A linha 80 é responsável pela impressão final.

O método baseia-se na junção de duas listas, ambas previamente submetidas a uma ordenação. A lista principal, composta de números randômicos desordenados, é dividida em duas outras listas (linhas 1010 e 1020). Observe que, na linha 1000, há um teste de saída para determinar o fim da recursão. Cada uma das listas sofre então uma ordenação parcial (linha 1030), sendo posteriormente reunidas. A sub-rotina irá chamar a si mesma (linha 1050) até que a ordenação esteja completa.

Apesar dos métodos de ordenação em BASIC não serem tão rápidos quanto os que empregam linguagem de máquina, este programa pode ser muito eficiente em tarefas menos extensas. Por exemplo, a ordenação de cem números, no Spectrum, leva mais ou menos 40 segundos; a ordenação tipo bolha nos faria esperar mais de uma hora.

APLICAÇÕES

A utilidade da recursão vai muito além do cálculo de fatoriais e outras funções matemáticas. Ela pode ser aplicada na programação de jogos e na construção dos mais complexos gráficos. Essa técnica também aparece em sistemas de inteligência artificial e no controle de robôs, além de ser empregada em processamento de linguagens (compiladores e interpretadores).

Os jogos de estratégia, como o xadrez ou os *wargames*, constituem uma outra área interessante de aplicação. O programa que apresentamos a seguir usa a recursão na simulação de um jogo clássico, *As Torres de Hanói*.

```
10 BORDER 6: PAPER 6: INK 0:
CLS
20 PRINT TAB 8; INVERSE 1;"□
TORRES DE HANOI□"
30 INPUT "DIGITE O NUMERO DE
ANEIS (2-9)□";N: IF N<2 OR N
>9 THEN  GOTO 30
35 DIM T(3)
36 LET A$="■□■": INK 2: FOR M
=21 TO 21-N STEP -1: PRINT AT
M,7;A$;AT M,15;A$;AT M,23;A$:
NEXT M: INK 0
37 PRINT  INK 2;AT 21,7;"■■■"
;AT 21,15;"■■■";AT 21,23;"■■
■"
38 FOR M=1 TO N: PRINT  INK 7
; PAPER 0;AT 20-T(1),8;N+1-M:
LET T(1)=T(1)+1: NEXT M
39 PRINT #1;AT 0,3;"QUALQUER
TECLA PARA COMECAR"
40 PAUSE 0: PRINT #1;AT 0,3;"
□□□□□□□□□□□□□□□□□□□□"
45 LET TT=2: LET TF=1: LET R=
3
50 GOSUB 90
70 PRINT AT 10,5;"NUMERO DE M
OVIMENTOS=□";2^N-1
80 STOP
90 IF N=0 THEN  RETURN
100 LET N=N-1: LET W=R: LET R=
TT: LET TT=W: GOSUB 90: LET W=
R: LET R=TT: LET TT=W
110 GOSUB 200
120 LET W=R: LET R=TF: LET TF=
W: GOSUB 90: LET W=R: LET R=TF
: LET TF=W
130 LET N=N+1: RETURN
200 PRINT AT 20-(T(TF)-1),TF*8
;"□"; INVERSE 1;AT 20-(T(TT)),
TT*8;N+1: LET T(TF)=T(TF)-1:
LET T(TT)=T(TT)+1
210 SOUND .01,TT*T(TT)*2
220 RETURN
```

```
10 CLS:DIM H(3)
20 PRINT @8,"TORRES DE HANOI":P
RINT
30 PRINT"NUMERO DE ANEIS (2-9)
?";
40 A$=INKEY$:IF A$<"2" OR A$>"9
" THEN 40
50 N=VAL(A$):H(0)=N:PRINT @64
60 FOR K=0 TO 8:FOR J=0 TO 2:PR
INT @165+K*32+J*9,CHR$(175)+" "
+CHR$(175);:NEXT J,K
70 FOR K=0 TO 2:PRINT @453+K*9,
STRING$(3,175);:NEXT
80 FOR K=1 TO N:POKE 1478-32*K,
49+N-K:NEXT
90 TT=1:TF=0:R=2
100 GOSUB 1000
110 PRINT @65,"NUMERO DE MOVIME
NTOS =";INT(2^N-1)
120 PRINT" QUALQUER TECLA PARA
RECOMECAR"
130 IF INKEY$="" THEN 130
140 RUN
1000 IF N=0 THEN RETURN
1010 N=N-1:W=R:R=TT:TT=W:GOSUB
1000:W=R:R=TT:TT=W
1020 POKE 1478+9*TF-32*H(TF),96
:H(TF)=H(TF)-1:H(TT)=H(TT)+1:PO
KE 1478+9*TT-32*H(TT),49+N
1030 W=R:R=TF:TF=W:GOSUB 1000:W
=R:R=TF:TF=W
1040 N=N+1:RETURN
```

```
10  HOME : DIM T(3)
20  HTAB (14): INVERSE : PRINT
"TORRE DE HANOI": NORMAL
30  PRINT : PRINT : INPUT "
    NUMERO DE ANEIS (2-9): ";N
40  IF N < 2 OR N > 9 THEN  RUN

50  FOR J = 1 TO 3: INVERSE : V
TAB (16): HTAB (J * 10): PRINT
J: NEXT : NORMAL
60  FOR I = 1 TO N: HTAB (10):
VTAB (16 - I): PRINT N - I: NEX
T I
70 T(1) = 15 - N:T(2) = 15:T(3)
 = 15
80 TT = 2:TF = 1:R = 3: GOSUB 1
10
90  HTAB (14): VTAB (20): PRINT
 "MOVIMENTOS=";2 ^ N - 1
100  END
110  IF N = 0 THEN  RETURN
120 N = N - 1:W = R:R = TT:TT =
 W: GOSUB 110:W = R:R = TT:TT =
 W
130  GOSUB 160
140 W = R:R = TF:TF = W: GOSUB
110:W = R:R = TF:TF = W
150 N = N + 1: RETURN
160 T(TF) = T(TF) + 1: HTAB (TF
 * 10): VTAB (T(TF)): PRINT " "

170  HTAB (TT * 10): VTAB (T(TT
)): PRINT N:T(TT) = T(TT) - 1
180  RETURN
```

```
10 KEY OFF:CLS:DIM T(3)
20 PRINT TAB(13)"TORRE DE HANOI
"
30 PRINT:PRINT:INPUT" NUMERO DE
 ANEIS (2-9): ";N
40 IF N<2 OR N>9 THEN RUN
50 FOR I=1 TO N:LOCATE 10,16-I:
PRINT N-I:NEXT I
60 T(1)=15-N:T(2)=15:T(3)=15
70 TT=2:TF=1:R=3:GOSUB 100
80 LOCATE 14,20:PRINT" MOVIMENT
OS=";2^N-1
90 END
100 IF N=0 THEN RETURN
110 N=N-1:W=R:R=TT:TT=W:GOSUB 1
00:W=R:R=TT:TT=W
120 GOSUB 150
130 W=R:R=TF:TF=W:GOSUB 100:W=R
:R=TF:TF=W
140 N=N+1:RETURN
150 T(TF)=T(TF)+1:LOCATE TF*10,
T(TF):PRINT"  "
160 LOCATE TT*10,T(TT):PRINT N:
T(TT)=T(TT)-1
170 RETURN
```

Quando o jogo começa, vários discos de diâmetros diferentes acham-se empilhados sobre a primeira de três varetas montadas sobre o tabuleiro. O objetivo é passar todos os discos para a segunda vareta, movendo um por vez e sem deixar um disco maior sobre um menor. A terceira vareta serve como apoio temporário durante o processo.

O computador mostrará uma simulação animada, pedindo, inicialmente, o número de discos que sairão da primeira vareta. A transferência destes será bastante rápida, o que dificulta a visualização dos movimentos. Caso queira observar melhor o processo, faça as modificações que se seguem. Para o Spectrum, introduza esta linha:

```
215 PAUSE 0
```

Para o TRS-Color, adicione este comando ao final da linha 1020:

```
:SOUND 50+H(TT)*10,12
```

No Apple e no TK-2000, acrescente:

```
175 FOR I=1 TO 800:NEXT
```

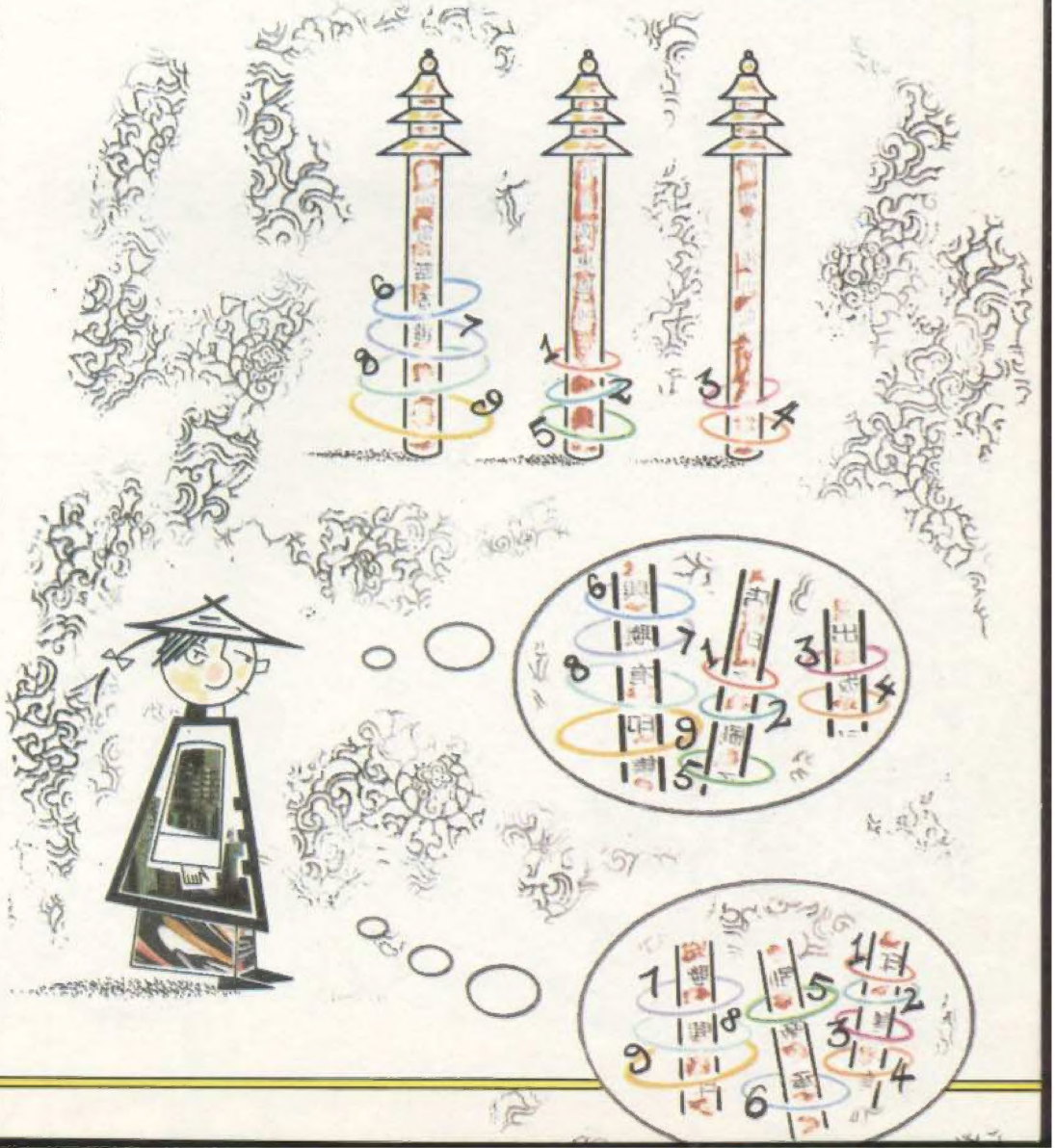
Para o MSX, adicione esta linha:

```
165 BEEP:FOR I=1 TO 800:NEXT
```

O programa é semelhante aos anteriores. A rotina recursiva executa cada movimento sucessivamente, até que todos os discos estejam na segunda vareta. No fim, o computador também exibe o total de movimentos realizados.

As Torres de Hanói **e seus nove discos.**

Apesar de poderosa, a recursão pode ser inconveniente quando há uma grande limitação de memória, roubando mais espaço e mais tempo do que o necessário. Porém, se não há tal impedimento e se em seu programa uma subrotina é chamada mais de duas vezes, procure aperfeiçoá-lo com essa técnica.



AVALANCHE: PONTOS GANHOS

Nem tudo é desastre no caminho de Willie. Para que não se desespere totalmente, às vezes é recompensado, conseguindo chegar ao topo da montanha e recuperar seu lanche.

Nenhum outro aventureiro conheceu tantos infortúnios quanto Willie. Nosso personagem já experimentou até a morte e a descida para o inferno. Merece, agora, uma recompensa: finalmente, ele conseguirá alcançar o topo da montanha e recuperar mais uma parte do lanche perdido, conquistando, assim, alguns pontos no placar.

A pequena rotina inicial executa a melodia da recompensa, coloca Willie no próximo nível, aumenta a velocidade do jogo e incrementa o placar.

```
 10 REM org 59788
 20 REM rwd ld de,523
 30 REM ld hl,806
 40 REM call 949
 50 REM ld a,(57344)
 60 REM inc a
 70 REM res 2,a
 80 REM ld (57344),a
 90 REM ld a,(58732)
100 REM dec a
110 REM ld (58732),a
120 REM ld a,2
130 REM ld b,5
140 REM call 59900
150 REM jp 58601
```

As três primeiras instruções, encarregadas da música, usam a rotina **BEEPER** no endereço 949 da ROM. Os parâmetros da duração e da tonalidade são fornecidos pelo método usual, através dos pares de registros DE e HL.

O NÍVEL DO JOGO

O nível de dificuldade do jogo é carregado da variável correspondente, em 57334, para o acumulador. O conteúdo do acumulador é em seguida incrementado. Precisamos, porém, impedir que seu valor seja maior do que 3, já que existem só quatro níveis de dificuldade. Para isso, usamos a instrução **res 2, a** que ajusta com 0 o bit 2 do acumulador. Quando o valor deste chega a 4, o bit 2 é colocado em 0. O conteúdo do acumulador volta, assim, a 0, trazendo o jogo para o primeiro nível.

O resultado dessa operação é armazenado no endereço 57334, onde será utilizado ao se ajustar o jogo.

A VELOCIDADE

O valor do atraso do jogo — que fica no endereço 58732 — é carregado no acumulador, decrementado e armazenado de volta em 58732. Aceleramos, assim, o jogo, uma vez que o tempo gasto pelo processador no laço de atraso da rotina principal diminui.

O atraso foi originalmente ajustado com 50. Como Willie não chegará à recompensa mais do que cinqüenta vezes, o jogo se tornará cada vez mais rápido — e, portanto, mais difícil. Cada vez que você enfrentar os mesmos quatro níveis, eles estarão mais rápidos.

CONTAGEM DE PONTOS

Por ter alcançado o prêmio, Willie recebe mais 500 pontos. Para isso, chamamos a rotina do escore no endereço 58900 e fornecemos os parâmetros nos registros A e B.

O valor 2 colocado em A especifica o segundo dígito a partir da esquerda — o das centenas —, que será incrementado. O valor 5 em B informa à rotina o número de vezes que esse dígito deve ser aumentado. Incrementando as centenas cinco vezes, acrescentamos 500 pontos ao escore.

Em seguida, o processador volta para a rotina de nova vida — rotulada **n1v** —, em 58601, e coloca nosso personagem na base da encosta.

O PLACAR

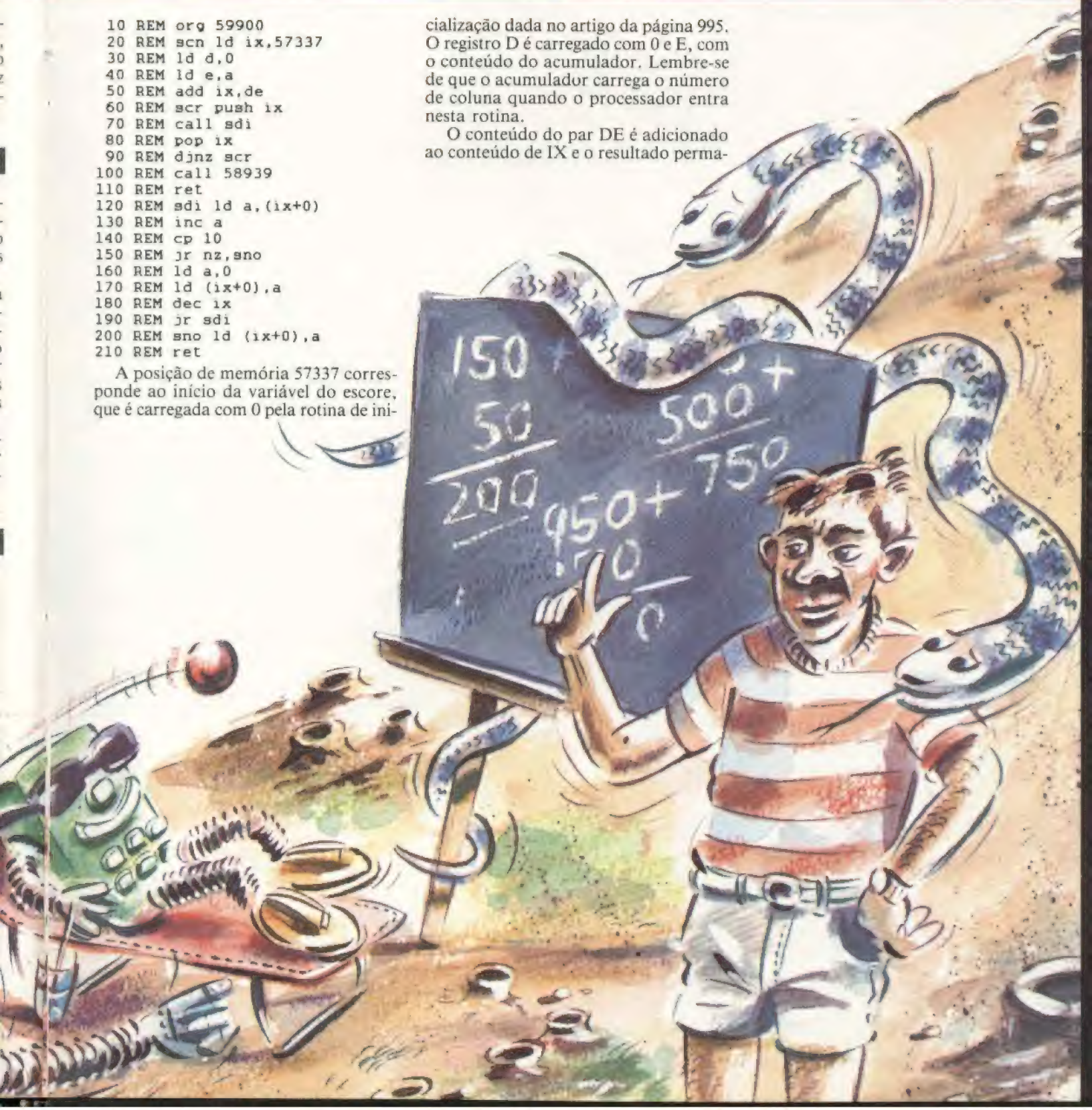
A pequena rotina apresentada a seguir acerta o placar de Willie quando ele alcança uma recompensa ou escala uma outra parte da encosta.

- NÍVEL DE DIFICULDADE
- FUNDO MUSICAL
- ACELERAÇÃO DO RITMO DO JOGO
- WILLIE ALCANÇA
- O PRÊMIO
- CONTAGEM DE PONTOS
- EXIBIÇÃO DO PLACAR
- ALTERAÇÃO DOS DÍGITOS
- DE VOLTA AO SOPÉ

```
 10 REM org 59900
 20 REM scn ld ix,57337
 30 REM ld d,0
 40 REM ld e,a
 50 REM add ix,de
 60 REM scr push ix
 70 REM call sdi
 80 REM pop ix
 90 REM djnz scr
100 REM call 58939
110 REM ret
120 REM sdi ld a,(ix+0)
130 REM inc a
140 REM cp 10
150 REM jr nz,sno
160 REM ld a,0
170 REM ld (ix+0),a
180 REM dec ix
190 REM jr sdi
200 REM sno ld (ix+0),a
210 REM ret
```

A posição de memória 57337 corresponde ao início da variável do escore, que é carregada com 0 pela rotina de inicialização dada no artigo da página 995. O registro D é carregado com 0 e E, com o conteúdo do acumulador. Lembre-se de que o acumulador carrega o número de coluna quando o processador entra nesta rotina.

O conteúdo do par DE é adicionado ao conteúdo de IX e o resultado perma-



nece em IX. Isso faz com que o apontador de dados percorra os dígitos armazenados a partir de 57337, até chegar ao especificado pelo conteúdo de A. A posição é temporariamente armazenada, colocando-se o conteúdo de IX na pilha. A rotina **sdi** é chamada.

ALTERANDO OS DÍGITOS

A rotina **sdi** é a que trata dos dígitos. Ela começa carregando o acumulador com o dígito apontado por IX. O deslocamento do 0 é necessário aqui por causa do formato da instrução.

O dígito é então incrementado e comparado com 10. Se ele for 10, será preciso incrementar também no próximo dígito. Caso contrário — ou seja, se o primeiro dígito ainda não foi incrementado até 10 —, a instrução **jr nz** manda o processador para o rótulo **sno**, que volta a armazenar o dígito incrementado na posição de onde veio.

Se o dígito que você está alterando foi incrementado até 10, o salto não ocorre e o processador continua com a próxima instrução. O valor 0 é armazenado no dígito apropriado. Em seguida, IX é decrementado, passando a apontar para o dígito imediatamente à esquerda. A instrução **jr sdi** manda então o processador de volta para **sdi**, iniciando novamente a rotina de incremento no próximo dígito.

Se este foi incrementado até 10, o processador continua no laço. Mas, cedo ou tarde, um dígito que não chegou a 10 será encontrado. O processador irá então para **sno**, armazenando o último dígito e retornando para o lugar da rotina **scr** de onde **sdi** foi chamada.

MAIS PLACAR

Quando o processador retorna da rotina, o apontador IX é recuperado da pilha outra vez. Você pode agora perceber por que precisamos armazená-lo ali: se o dígito foi incrementado até 10, **sdi** terá deslocado o apontador IX para o dígito seguinte; caso você tente incrementá-lo mais uma vez, irá alterar o dígito errado.

O laço desta parte da rotina é fechado por uma instrução **djnz**, que decrementa o conteúdo de B até zerá-lo. Como você deve se lembrar, B carrega o número de vezes que o dígito do escore tem que ser incrementado quando o processador entra na rotina.

O processador continua no laço atualizando o escore o número de vezes inicialmente especificado por B. Quando

B chega a 0, a rotina de impressão em 58939 é chamada, imprimindo o novo escore na tela.

Em seguida, o processador retorna.

Esta pequena rotina executa a melodia da recompensa, coloca Willie no próximo nível, acelera o jogo e acrescenta pontos no placar.

```
10   ORG 20721
20   RWD LDA #255
30   LDX #150
40   JSR SOUND
50   LDA 18238
60   INCA
70   ANDA #3
80   STA 18238
90   DEC DLL+1
100   LDB #5
110   LDA #3
120   JSR SCI
130   LBRA NLV
140   SCI EXG A,B
150   SCT LDX #18240
160   ABX
170   PSHS A,X
180   JSR SDI
190   PULS A,X
200   DECA
210   BNE SCT
220   JSR PRSC
230   RTS
240   SDI LDA,X
250   INCA
260   CMPA #10
270   BNE SNO
280   CLR ,X
290   LEAX -1,X
300   BRA SDI
310   SNO STA ,X
320   RTS
330   SOUND PSHS A
340   LDA $FF01
350   ANDA #247
360   STA $FF01
370   LDA $FF03
380   ANDA #247
390   STA $FF03
400   LDA $FF23
410   ORA #8
420   STA $FF23
430   ORCC #$50
440   PULS A
450   PSHS X
460   LDB #252
470   SBN STB $FF20
480   SC LEAX -1,X
490   BNE SC
500   LDX,S
510   CLR $FF20
520   SD LEAX -1,X
530   BNE SD
540   LDX ,S
550   DECA
560   BNE SBN
570   ANDCC #$AF
580   PULS X
590   RTS
600   CLICK LDX #98
610   LDA #4
620   JSR SOUND
630   RTS
640   DLL EQU $51ED
650   NLV EQU $4BF7
660   PRSC EQU $4C77
```

As três primeiras instruções, encarregadas da música, usam a rotina **SOUND**, chamada pela linha 40. Os parâmetros da duração e da tonalidade são fornecidos pelo método usual, através de números carregados em A e X.

NÍVEL DE DIFICULDADE

A é carregado com o conteúdo de 18238, a posição de armazenamento do nível de dificuldade. Esse valor é incrementado e a operação AND é feita com o número 3. Apagamos, assim, os seis bits mais significativos e evitamos que o nível do jogo ultrapasse o valor 3. O resultado é armazenado de volta na posição 18238.

Em seguida, a variável na posição de memória $51EE é decrementada e o jogo se torna um pouco mais rápido.

Para atualizar o escore, carrega-se 5 em B e 3 em A; a rotina **SCI**, fornecida a seguir, é então chamada. B carrega o número de vezes que o dígito será incrementado e A identifica esse dígito. Desta vez, portanto, acrescentamos 500 pontos ao escore.

Finalmente, o processador volta para a rotina **NLV**, que irá colocar o próximo nível na tela.

CÁLCULO DO PLACAR

Para serem utilizados na próxima rotina, os conteúdos dos registradores A e B precisam ser trocados. Isso é feito pela instrução **EXG**. X é então carregado com 18240, o endereço inicial dos valores do escore.

ABX adiciona a X o conteúdo de B — que é o número do dígito a ser incrementado. Em outras palavras, essa instrução desloca o apontador em X, que estava no início dos dados do escore, para a posição do dígito que se pretende incrementar. O conteúdo de A (o número de vezes que o dígito será incrementado) e o de X (a posição de memória desse dígito) são colocados na pilha de máquina. Em seguida, o processador salta para a sub-rotina **SDI** que executa o incremento. Os conteúdos de A e X são recuperados da pilha.

A é decrementado e, se não tiver chegado a 0, a instrução **BNE SCT** manda o processador continuar no laço. A rotina de incremento do dígito é executada A vezes, incrementando o dígito apropriado a cada volta do laço.

Com A reduzido a 0 e o escore acertado, o processador sai do laço e vai para a rotina **PRSC**, que imprime o resultado na tela. Depois disso, o processador encontra **RTS** e retorna para a rotina principal.

ALTERANDO OS DÍGITOS

A é carregado com o conteúdo da posição de memória apontada por X. Este corresponde ao número contido no dígito apropriado do escore.

O número é incrementado e o resultado é comparado com 10. Se ainda não.

for igual a 10, a instrução **BNE** coloca o processador no rótulo **SNO**, que volta a armazenar o dígito que foi incrementado no endereço apontado por X. O processador retorna.

Se o resultado for igual a 10, o dígito apontado por X é ajustado com 0 — ou seja, é limpo. X é então decrementado, fazendo o apontador se mover para o dígito imediatamente à esquerda.

BRA SDI manda o processador de novo para o laço de incremento de dígito. Depois que o próximo dígito é incrementado, verifica-se se ele ultrapassou o valor 9. Encontrando um dígito que não exceda o valor máximo, o processador o armazena na posição apontada por X. Observe que, ao voltar da rotina de escore, o processador recupera o valor de X da pilha.

A rotina **SOUND** trabalha exatamente do mesmo modo que a rotina encarregada de executar *Greenleaves*. Porém, toca apenas uma vez e usa a pilha como fonte de dados.

Depois de **SOUND**, há uma outra pequena rotina chamada **CLICK**. Ela executa a trilha sonora da caminhada de Willie, carregando os parâmetros em X e A e chamando a rotina **SOUND**.

Para testar a rotina, use as teclas M e N e este programa em BASIC.

```
10 POKE 3000,87
20 EXEC 19426
30 EXEC 19902
40 GOTO 30
```

Esta pequena rotina coloca Willie no nível seguinte, aumenta a velocidade do jogo e acrescenta os pontos ganhos ao placar.

```
10    org 55506
20    ld a,(-5228)
30    inc a
40    res 2,a
50    ld (-5228),a
60    ld a,(54133)
70    dec a
80    ld (54133),a
90    ld a,2
100    ld b,5
110    call 55532
120    jp 53888
```

O NÍVEL DE DIFICULDADE

O nível de dificuldade do jogo é transferido da variável equivalente em – 5228 para o acumulador. A seguir, o conteúdo desse é incrementado. Como existem apenas quatro níveis no jogo, o valor dessa variável não deve ultrapassar 3. Para evitar que isso ocorra, utilizamos a instrução **res 2,a**, que ajusta com 0 o bit 2 do acumulador. Assim, quando o valor incrementado chega a 4, o bit 2 é ajustado com 0. O conteúdo do acumulador volta, então, a 0, colocando o jogo novamente no primeiro nível.

O resultado dessa operação é armazenado em – 5228, onde o novo valor será utilizado no controle do jogo.

VELOCIDADE

O atraso do jogo em 54133 é colocado no acumulador, decrementado e armazenado de volta em 54133. Isso acelera o jogo, já que o tempo gasto pelo processador no laço de atraso da rotina principal diminui.

Como você poderá verificar na última rotina da série *Avalanche*, esse atraso foi originalmente ajustado com 50. Assim, sempre que aumenta o número de pontos no placar, o jogo se torna mais rápido e difícil. Cada vez que você enfrentar os mesmos quatro níveis, a velocidade será maior.

CONTAGEM DOS PONTOS

Por ter alcançado o prêmio, Willie recebe mais 500 pontos. Para isso, chamamos a rotina de contagem e fornecemos os parâmetros nos registros A e B. O valor 2 colocado em A especifica o segundo dígito a partir da esquerda — o das centenas —, que será incrementado. O valor 5 em B informa à rotina o número de vezes que esse dígito deve ser aumentado. Incrementando as centenas cinco vezes, acrescentamos 500 pontos ao escore. Em seguida, o processador vai para a rotina principal do jogo, em 53888, e coloca Willie na base da encosta.

O PLACAR

A pequena rotina que se segue acerta os pontos de Willie quando ele alcança uma recompensa ou escala uma parte da encosta.

```
130    org 55532
140    po ld hl,-5219
150    ld d,0
160    ld e,a
170    add hl,de
180    pd push hl
190    call sd
200    pop hl
210    djnz pd
220    call 54023
230    ret
240    sd ld a,(hl)
250    inc a
260    cp 10
270    jr nz,sn
280    ld a,0
290    ld (hl),a
300    dec hl
310    jr sd
320    sn ld (hl),a
330    ret
340    end
```

A posição de memória – 5219 corresponde ao endereço inicial das variáveis que contêm os dígitos do escore, carregadas com 0 pela rotina de inicialização dada em artigo anterior. O registro D é carregado com 0 e o registro E, com o conteúdo do acumulador. Lembre-se de que o acumulador carrega o número do dígito quando o processador entra nesta rotina.

O conteúdo do par DE é somado em HL. A operação faz o apontador de dígitos percorrer as variáveis onde seus valores estão armazenados, a partir de – 5219, até o dígito indicado pelo conteúdo de A. O conteúdo de HL é colocado na pilha, onde ficará temporariamente armazenado, e **sd** é chamada.

ALTERANDO OS DÍGITOS

A rotina **sd**, que trata dos dígitos, começa carregando o acumulador com o

valor do dígito apontado por HL. Este é incrementado e comparado com 10. Se for igual a 10 — ou seja, se ultrapassou o maior valor de um dígito decimal, 9 —, será preciso incrementar o próximo dígito. Se o dígito analisado ainda não ultrapassou 9, a instrução **jr nz** manda o processador para o rótulo **sn**, que armazena seu valor na variável correspondente.

Quando o dígito que está sendo alterado chega a 10, o salto não ocorre e o processador continua com a próxima instrução. O valor 0 é armazenado nesse dígito e o par HL é decrementado, passando a apontar para o dígito imediatamente à esquerda. A instrução **jr sd** manda o processador de volta para **sd** e a rotina de incremento recomeça para o próximo dígito.

Se este também já ultrapassou 9, o processador continua no laço. Mas, cedo ou tarde, um dígito que não chegou a 10 será encontrado. O processador irá então para **sn**, armazenando o último dígito e retornando para o lugar da rotina **pd** de onde **sd** foi chamada.

MAIS PLACAR

Quando o processador retorna da rotina, o apontador HL é recuperado da pilha. Você pode agora perceber por que precisamos armazená-lo ali: se algum dígito foi incrementado até 10, a rotina **sd** terá deslocado o apontador. Caso não tivéssemos armazenado o valor inicial na pilha, desta vez estaríamos alterando o dígito errado.

O laço é fechado por uma instrução **djnz**, que decrementa o conteúdo de B até zerar esse registro. Como você deve se lembrar, B carrega o número de vezes que o dígito do escore tem que ser incrementado quando o processador entra nesta rotina.

O processador continua no laço **pd** atualizando o escore o número de vezes inicialmente especificado por B. Quando B chega a 0, a rotina de impressão dos dígitos em 54023 é chamada, imprimindo o novo escore na tela.

Em seguida, o processador retorna.

OS DADOS VÃO ROLAR

INPUT apresenta, finalmente, um jogo de computador projetado para vários participantes. Reúna a família e os seus amigos e role os dados nesta brincadeira de sorte e de perícia.

Até aqui, a maioria dos jogos de *INPUT* — sejam os de aventura, estratégia ou videogame — basearam-se no confronto computador-usuário. A maior dificuldade nesse tipo de disputa estava, como vimos, na criação de regras e situações que simulassem desafio ou, no caso dos jogos de estratégia, na transformação do computador em um adversário inteligente e de bom nível. O jogo que apresentamos neste artigo é diferente: o confronto pode se dar entre até seis jogadores. O computador não participa como adversário — cabe-lhe apenas conferir as regras e manter o placar. A máquina evita, assim, que alguém trapaceie, e, o que é melhor, libera os jogadores da contagem do placar, possibilitando que se concentrem só nas questões estratégicas.

Este jogo não passa de uma versão computadorizada do famoso *Yatch*, ou pôquer com dados. Combinando sorte e estratégia, é muito absorvente, mas suas regras são simples, parecidas com as do pôquer. Cada participante tem direito a jogar cinco dados de uma só vez (no nosso caso, o computador simula os dados rolando). Se o jogador não estiver satisfeito com o resultado obtido, poderá lançar os dados mais duas vezes (num total de três jogadas). Na segunda e terceira jogadas, é permitido escolher quais dentre os cinco dados serão lançados novamente, para que se consiga a melhor "mão" possível. Depois do terceiro lançamento, o jogador deve indicar o grupo do placar no qual sua combinação de dados vai ficar (cada grupo poderá ser usado apenas uma vez). A partida continua, então, com o próximo jogador.

Os grupos de placar são:

GRUPO	PONTOS
UM	soma dos 1 obtidos
DOIS	soma dos 2 obtidos
.	.
.	.
.	.
SEIS	soma dos 6 obtidos
FULL HOUSE	soma dos cinco dados
CURTO	15
LONGO	30
MISTO	soma dos cinco dados
YATCH	50

Um CURTO é uma combinação de quatro dados em seqüência (1,2,3,4 ou 2,3,4,5) e um LONGO, uma combinação de cinco dados também em seqüência (1,2,3,4,5 ou 2,3,4,5,6). Um FULL HOUSE compõe-se de uma trinca e um par (5,5,5,1,1, por exemplo), enquanto um MISTO, como seu próprio nome diz, aceita qualquer tipo de combinação de dados. O YATCH, finalmente, é o grupo formado por cinco dados iguais.

O resultado obtido numa jogada deve ser colocado num dos grupos. Lembre-se de que só se pode indicar cada grupo uma vez. Para selecionar o grupo e executar a escolha usam-se as setas e a barra de espaço.

Muitas vezes não é possível obter uma combinação de dados que satisfaça um dos grupos vazios (os grupos cheios não podem ser utilizados novamente). Neste caso, a única opção é sacrificar um dos grupos vazios — de preferência, um dos que rendem menos pontos —, eliminando-o. A melhor estratégia, portanto, é preencher primeiro os grupos que rendem mais pontos — que são também os mais difíceis — evitando-se que venham a ser sacrificados.

O programa está dividido em três partes principais: rotina de inicialização, laço-mestre e rotinas chamadas pelo laço-mestre.

INICIALIZAÇÃO

Esta parte do programa configura os UDG (gráficos definidos pelo usuário) que representarão os dados na tela (menos nos micros das linhas Apple e TK-2000), inicializa as variáveis e armazena os nomes dos jogadores.

```
10 CLS:X$=CHR$(13)+CHR$(10):DIM
D$(6,4)
20 FORK=1TO6:FORJ=1TO3:FORL=1TO
3:READA:D$(K,J)=D$(K,J)+CHR$(1)
+CHR$(219-145*A):NEXT:NEXT:NEXT
30 DATA 0,0,0,0,1,0,0,0,0
40 DATA 1,0,0,0,0,0,0,0,1
50 DATA 1,0,0,0,1,0,0,0,1
60 DATA 1,0,1,0,0,0,1,0,1
70 DATA 1,0,1,0,1,0,1,0,1
80 DATA 1,0,1,1,0,1,1,0,1
85 LOCATE7,10:PRINT"Quantos jog
adores (1-6)?";
90 A$=INKEY$:IFA$<"1" OR A$>"6"
THEN90
100 PRINTA$:NP=VAL(A$):CLS
110 FORN=1TONP:PRINT"nome do jo
gador";N;:INPUTN$(N):NEXT
120 CLS:CL$=STRING$(35," ")
130 DIMO(NP,12),P(NP,12),S(NP,1
0)
```

```
10  HOME : DIM A$(12)
20 CL$ = "

30  FOR T = 1 TO 6: READ A$:DC$
(T) = A$: NEXT
40  FOR T = 1 TO 12
50  READ A$:A$(T) = A$ +  LEFT$
 (".............",12 -  LEN (A$)
): NEXT
60  VTAB (10): PRINT  TAB( 10);
"QUANTOS JOGADORES (1-6)?";
70  GET A$: IF A$ < "1" OR A$ >
 "6" THEN 70
80 NP =  VAL (A$): PRINT NP: DI
M N$(NP),SC(NP),O(NP,12),P(NP,1
2),S(NP,10)
90  HOME : FOR N = 1 TO NP: PRI
NT  TAB( 1);"NOME DO JOGADOR ";
N;" = ";
100  INPUT N$(N): NEXT
1170  DATA  <1>,<2>,<3>
1180  DATA  <4>,<5>,<6>
1190  DATA  UNS,DOIS,TRES,QUAT
ROS,CINCOS,SEIS,4 IGUAIS
1200  DATA  CASA CHEIA,CURTO,L
ONGO,MISTO,YATCH
```

```
20 LET Q$=".............": LET
Z$="              ": DIM C(13):
FOR N=1 TO 13: READ C(N):
NEXT N: DIM T(5): DIM R(5):
DIM D(5)
30 FOR N=USR "A" TO USR "G"+7
: READ A: POKE N,A: NEXT N
40 DATA 2,3,4,5,6,7,11,14,17,
20,23,25,27
50 DATA 0,0,0,24,24,0,0,0
60 DATA 0,6,6,0,0,96,96,0
70 DATA 3,3,0,24,24,0,192,192
80 DATA 0,102,102,0,0,102,102
,0
90 DATA 195,195,0,24,24,0,195
,195
```

- AS REGRAS DO JOGO
- A ESTRATÉGIA
- GRUPOS DE PLACAR
- UDG DOS DADOS
- O LAÇO-MESTRE
- ROLANDO OS DADOS
- O PLACAR
- IMPRESSÃO
- DO ESQUELETO
- RESULTADO FINAL


```
100 DATA 102,102,0,102,102,0,
102,102
110 DATA 0,24,48,96,255,96,48,
24
120 PRINT AT 10,13;"YATCH":
INK 1: PRINT AT 12,7;"QUANTOS
JOGADORES"''TAB 11;"(1 A 6)"
130 INPUT NP: LET NP=INT (NP):
IF NP<1 OR NP>6 THEN  GOTO 130
140 DIM O(NP,12): DIM P(NP,12)
: DIM S(NP,5): DIM N$(NP,6):
DIM Q(NP)
150 FOR N=1 TO NP: CLS : PRINT
AT 8,5;"JOGADOR ";(N);"."''TAB
5;"QUAL E SEU NOME ?": INPUT
W$: IF LEN W$>6 THEN  LET W$=
W$( TO 6)
160 LET N$(N)=Z$( TO 3-(LEN W$
)/2)+W$: NEXT N
```

```
10 CLS:X$=CHR$(13):DIM D$(6,4)
20 FOR K=1 TO 6:FOR J=1 TO 3:FO
R L=1 TO 3:READ A:D$(K,J)=D$(K,
J)+CHR$(128+65*A):NEXT
30 D$(K,J)=D$(K,J)+CHR$(133):NE
XT
40 D$(K,J)=STRING$(3,131)+CHR$(
135):NEXT
50 DATA 0,0,0,0,1,0,0,0,0,1,0,0
,0,0,0,0,1
60 DATA 1,0,0,0,1,0,0,0,1,1,0,1
,0,0,0,1,0,1
70 DATA 1,0,1,0,1,0,1,0,1,1,0,1
,1,0,1,1,0,1
80 PRINT:PRINT"QUANTOS JOGADORE
S (1-6) ?";
90 A$=INKEY$:IF A$<"1" OR A$>"6
" THEN 90
100 PRINT A$:NP=VAL(A$):CLS
110 FOR N=1 TO NP:PRINT @65,"JO
GADOR";N:PRINT" QUAL E SEU NOME
 ?":INPUT N$(N)
120 CLS:NEXT
130 DIM O(NP,12),P(NP,12),S(NP,
10)
```

O LAÇO-MESTRE

A estrutura do jogo é bem simples, e consiste nestas poucas linhas:

```
140 FORR=1TO5:FORI=1TO12:FORN=1
TONP
150 CLS:Y=2:GOSUB980:GOSUB190
```

```
170 CLS:GOSUB350:CLS:NEXTN,I
180 CLS:GOSUB990:NEXTR:END
980 LOCATE19-LEN(N$(N))/2,Y:PRI
NTN$(N):RETURN
```

```
140  FOR R = 1 TO 5: FOR I = 1
TO 12: FOR N = 1 TO NP
150  HOME :Y = 2: GOSUB 980: GO
SUB 190
170  HOME : GOSUB 350: HOME : N
EXT N,I
180  HOME : GOSUB 990: NEXT R:
END
980  HTAB (19 -  LEN (N$(N)) /
2): VTAB (Y): PRINT N$(N): RETU
RN
```

```
170 FOR R=1 TO 5: FOR I=1 TO
12: FOR N=1 TO NP
180 BORDER 4: INK 0: PAPER 4:
CLS : PRINT AT 3,13;N$(N)
190 FOR M=5 TO 27: PRINT
PAPER 0;AT 5,M;" ";AT 19,M;" "
: NEXT M
200 FOR M=6 TO 18: PRINT
PAPER 0;AT M,5;" ";AT M,27;" "
: NEXT M
210 GOSUB 240: PAUSE 0: GOSUB
 430
230 NEXT N: NEXT I: GOSUB 1290
: NEXT R: STOP
```

```
140 FOR R=1 TO 5:FOR I=1 TO 12:
FOR N=1 TO NP
```

```
150 CLS:W=6:Y=2:GOSUB 980:GOSUB.
 190
160 SOUND 50,3:FOR E=1 TO 800:N
EXT
170 CLS:GOSUB 350:CLS:NEXT N,I
180 CLS:GOSUB 990:NEXT R:END
9870 PRINT @Y*32+W-((LEN(N$(N))
)/2),N$(N):RETURN
```

O laço-mestre compõe-se de três laços encadeados (um dentro do outro) que controlam o jogo. **R** é o número da partida; **I**, o número de jogadas dentro de uma partida e **N**, o número de jogadores. As rotinas chamadas por esse laço fazem os dados rolar e apresentam os placares parcial e final. Todos os micros, menos o Spectrum, chamam uma pequena rotina para centralizar os nomes na tela. As rotinas chamadas pelo laço-mestre também são subdivididas, como veremos a seguir.

OS DADOS VÃO ROLAR

A primeira dessas rotinas rola os dados, exibe-os na tela e chama duas outras rotinas.

Acrescente as linhas que se seguem à parte do programa já digitada:

```
190 T=1:FORD=1TO5:T(D)=INT(RND(
-TIME)*6)+1:NEXT
200 LOCATE0,T*2+3:PRINT"JOGO";T
.
210 GOSUB970:IFT=3 THEN 310
220 C=1:FOR D=1 TO 5
230 LOCATE9+D*4,9:PRINT"?"
240 A$=INKEY$:IFA$<>"N"ANDA$<>"
n"ANDA$<>"s"ANDA$<>"S"THEN240
250 IF A$="N"ORA$="n"THENPLAY"O
3L64CB":GOTO270
260 PLAY"O6L64CB":TR(C)=T(D):C=
C+1
270 LOCATE0,9:PRINTCL$:NEXTD
280 IFC=6THENGOSUB340:RETURN
290 FORD=C TO 5:TR(D)=INT(RND(-
TIME)*6)+1:NEXTD
300 GOSUB340
310 T=T+1
320 IFT<>4THEN200
330 FORE=1TO700:NEXT:RETURN
340 FORD=1TO5:T(D)=TR(D):NEXTD:
RETURN
970 FORD=1TO5:FORG=1TO4:LOCATED
*4+8,G+4:PRINTD$(T(D),G);:NEXTG
,D:RETURN
```

```
190 T = 1: FOR D = 1 TO 5:T(D)
=  INT ( RND (1) * 6) + 1: NEXT

200  VTAB (T * 2 + 3): PRINT "J
OGO ";T;
210  GOSUB 970: IF T = 3 THEN 3
10
220 C = 1: FOR D = 1 TO 5
230  HTAB (9 + D * 4): VTAB (9)
: PRINT "?";
240  GET A$: IF A$ <  > "S" AND
 A$ <  > "N" THEN 240
250  IF A$ = "N" THEN 270
260 TR(C) = T(D):C = C + 1
270  HTAB (9 + D * 4): VTAB (9)
: PRINT " ": NEXT D
280  IF C = 6 THEN  GOSUB 340:
RETURN
290  FOR D = C TO 5:TR(D) =  IN
T ( RND (1) * 6) + 1: NEXT
300  GOSUB 340
310 T = T + 1
320  IF T <  > 4 THEN 200
330  FOR E = 1 TO 700: NEXT : R
ETURN
340  FOR D = 1 TO 5:T(D) = TR(D
): NEXT : RETURN
970  FOR D = 1 TO 5: HTAB (D *
4 + 8): VTAB (6): PRINT DC$(T(D
));: NEXT : RETURN
```

```
240 LET T=1: FOR D=1 TO 5: LET
T(D)=INT (RND*6)+1: NEXT D
250 PRINT AT 6+T*3,7;"JOGO ";T
260 GOSUB 1180
270 IF T=3 THEN  GOTO 390
280 LET C=1: FOR D=1 TO 5
290 PRINT AT 7+T*3,16+D*2;"?"
300 FOR J=1 TO 50: NEXT J
310 LET A$=INKEY$: IF A$=""
THEN  GOTO 310
320 IF A$="N" THEN  SOUND .1,
10: GOTO 360
330 IF A$<>"S" THEN  GOTO 310
340 SOUND .1,30
350 LET R(C)=T(D): LET C=C+1
360 PRINT AT 7+T*3,16+D*2;" ":
NEXT D
370 IF C=6 THEN  GOSUB 420:
LET T=4: GOTO 400
380 FOR D=C TO 5: LET R(D)=INT
(RND*6)+1: NEXT D: GOSUB 420
390 LET T=T+1
400 IF T<>4 THEN  GOTO 250
410 RETURN
420 FOR D=1 TO 5: LET T(D)=R(D
): NEXT D: RETURN
1180 FOR D=1 TO 5: PRINT  PAPER
 2; INK 6; BRIGHT 1;AT 6+T*3,16
+D*2;CHR$ (143+T(D)): PAUSE 2:
SOUND .01,RND*40: NEXT D: RETUR
N
```

```
140 FOR R=1 TO 5:FOR I=1 TO 12:
FOR N=1 TO NP
150 CLS:W=6:Y=2:GOSUB 980:GOSUB
 190
160 SOUND 50,3:FOR E=1 TO 800:N
EXT
170 CLS:GOSUB 350:CLS:NEXT N,I
180 CLS:GOSUB 990:NEXT R:END
190 T=1:FOR D=1 TO 5:T(D)=RND(6
):NEXT
200 PRINT @64*T+64,"LANCE:";T;
210 GOSUB 970:IF T=3 THEN 310
220 C=1:FOR D=1 TO 5
230 PRINT @288,TAB(9+D*4)"?"
240 A$=INKEY$:IF A$<>"N" AND A$
<>"S" THEN 240
250 IF A$="N" THEN SOUND 10,1:G
OTO 270
260 SOUND 100,1:TR(C)=T(D):C=C+
1
270 NEXT D:PRINT @288
280 IF C=6 GOSUB 340:RETURN
290 FOR D=C TO 5:TR(D)=RND(6):N
EXT D
300 GOSUB 340
310 T=T+1
320 IF T<>4 THEN 200
330 RETURN
340 FOR D=1 TO 5:T(D)=TR(D):NEX
T D:RETURN
970 FOR D=1 TO 5:FOR G=1 TO 4:P
RINT @136+G*32+D*4,D$(T(D),G);:
NEXT G,D:RETURN
9870 PRINT @Y*32+W-((LEN(N$(N))
)/2),N$(N):RETURN
```

No primeiro lançamento, utilizam-se os cinco dados. O resultado é exibido na tela por meio da rotina da linha 970 (1180, no Spectrum). O jogador escolhe então quais são os dados "bons" e quais são os "ruins", indicando os primeiros com um S e os segundos com um N. Ele terá mais duas chances de lançar os dados ruins, devendo teclar S(im) para os dados que quer "segurar" e N(ão) para os demais.

Os números dos cinco dados iniciais são armazenados na matriz **T( )**. Depois do primeiro lançamento, os dados que "seguramos" são guardados temporariamente na matriz **R( )**, que é preenchida com outros valores randômicos até ficar com cinco números. Esses valores são levados de volta para a matriz **T( )** por meio da rotina da linha 340 (420, no Spectrum). Os resultados são exibidos na tela mais uma vez e o processo se repete no próximo lançamento.

O PLACAR

A maior parte da rotina dedica-se aos cálculos do placar e à verificação de entradas.

```
350 CLS
360 Y=0:GOSUB980:PRINTSTRING$(3
0,205);" PLACAR"
370 PRINT"UNS.....:"X$"DOIS
.....:"X$"TRES.....:"X$"Q
UATROS.....:"X$"CINCOS.....:"X
$"SEIS.....:"
380 PRINT"4 IGUAIS....:"X$"CASA
 CHEIA..:"X$"CURTO.....:"X$"L
ONGO.....:"X$"MISTO.....:"X
$"YATCH.....:"X$
390 PRINT"TOTAL.....:";
400 GOSUB460
410 GOSUB490:GOSUB460
420 LOCATE22,5:PRINT"<qualquer
tecla>"
440 A$=INKEY$:IFA$=""THEN440
```

```
450 RETURN
460 FORD=1TO12:IFP(N,D)=1 THENL
OCATE16,1+D:PRINT"X";
470 IFO(N,D)<>0THENLOCATE15,1+D
:PRINTO(N,D);
480 NEXTD:C=0:FORD=1TO12:C=C+O(
N,D):NEXT:LOCATE15,15:PRINTC;:R
ETURN
490 LOCATE2,20:PRINT"TEMPO"R;:L
OCATE1,21:PRINT"JOGADA"I;
500 LOCATE0,17:PRINT"RESULTADO
=";
510 FORD=1TO5:FORG=1TO4:LOCATE9
+4*D,15+G:PRINTD$(T(D),G);:NEXT
G,D
520 LOCATE16,20:PRINT"ESCOLHA O
 GRUPO";
530 A=1
540 LOCATE13,1+A:PRINTCHR$(60);
550 B$=INKEY$:IFB$<>" "ANDB$<>C
HR$(30)ANDB$<>CHR$(31)THEN550
560 IFB$=" "THEN620
570 LOCATE13,1+A:PRINT" ";
580 IFB$=CHR$(30)ANDA>1THENA=A-
1
590 IFB$=CHR$(31)ANDA<12THENA=A
+1
600 PLAY"O6A"
610 GOTO540
620 LOCATE13,1+A:PRINT" ";:IFP(
N,A)<>0THEN950
630 IFA>6THEN700
640 C=0
650 FORD=1TO5:IFT(D)=A THENC=C+
1
660 NEXTD
670 O(N,A)=C*A
680 P(N,A)=1
690 RETURN
700 IFA=11THENFORD=1TO5:O(N,11)
=O(N,11)+T(D):NEXT:P(N,11)=1:RE
TURN
710 FORD=1TO5:D(D)=0:NEXT:B=0:F
ORE=1TO6:C=0:FORD=1TO5:IFT(D)=E
 THENC=C+1
720 NEXTD:IFC<>0THENB=B+1
730 NEXTE
740 G=1:FORF=1TO6:GOSUB1140:IFC
<>0THEND(G)=F:G=G+1
750 NEXTF
760 P(N,A)=1:A=A-6:ONA GOTO770,
810,830,870,960,890
770 IFB>2THEN900:IFB=1GOSUB1160
:O(N,7)=C*4:RETURN
780 F=1
790 GOSUB1140:F=F+1:IFC<>4ANDF<
>7THEN790ELSEIFC<4THEN900
800 O(N,7)=4*(F-1):RETURN
810 IFB<>2THEN900ELSEF=D(1):GOS
UB1140:IFC=3THEN820ELSEF=D(2):G
OSUB1140:IFC<>3THEN900
820 FORD=1TO5:O(N,8)=O(N,8)+T(D
):NEXT:RETURN
830 IFB<>4THEN850ELSEGOSUB1160:
IFC<>18ANDC<>10ANDC<>14 OR(C=14
ANDD(4)=6)THEN900
840 O(N,9)=15:RETURN
850 IFB<>5THEN900ELSEGOSUB1160:
IFC<>20ANDC<>15ANDC<>16ANDC<>19
THEN900
860 GOTO840
870 IFB<>5THEN900ELSEGOSUB1160:
IFC<>20ANDC<>15THEN900
880 O(N,10)=30:RETURN
890 IFB<>1THEN900ELSEO(N,12)=50
:PLAY"GFG":FORE=1TO700:NEXT:RET
URN
900 PLAY"O3GGG":LOCATE22,5:PRIN
T"ilegal. ELIMINA?";
910 A$=INKEY$:IFA$<>"S"ANDA$<>"
s"ANDA$<>"N"ANDA$<>"n"THEN910
920 LOCATE22,5:PRINTSTRING$(17,
32);
930 IFA$="N"ORA$="n"THENP(N,A+6
)=0:GOTO530
940 P(N,A+6)=1:RETURN
950 PLAY"O3CFD":LOCATE22,5:PRIN
T"GRUPO OCUPADO";:FORE=1TO700:N
EXT:LOCATE22,5:PRINTSTRING$(14,
32);:GOTO530
960 RETURN
1140 C=0:FORD=1TO5:IFT(D)=F THE
NC=C+1
1150 NEXTD:RETURN
1160 C=0:FORD=1TOB:C=C+D(D):NEX
TD:RETURN
```

```
350  HOME
360 Y = 1: GOSUB 980: INVERSE :
 PRINT ">>>>>>>>>>>>>>>>>>>>>>>>
>>> PLACAR <<<"
370  NORMAL : PRINT : FOR T = 1
 TC 12: PRINT A$(T): NEXT
380  PRINT : INVERSE : PRINT "T
OTAL.....": NORMAL
400  GOSUB 460
410  GOSUB 490: GOSUB 460
420  HTAB (22): VTAB (8): PRINT
 "<QUALQUER TECLA>";
440  GET A$: IF A$ = "" THEN 44
0
450  RETURN
460  FOR D = 1 TO 12: IF P(N,D)
 = 1 THEN  HTAB (15): VTAB (3 +
 D): PRINT "X";
470  IF O(N,D) <  > 0 THEN  HTA
B (15): VTAB (3 + D): PRINT O(N
,D);
```

```
480  NEXT D:C = 0: FOR D = 1 TO
 12:C = C + O(N,D): NEXT
485  HTAB (15): VTAB (17): PRIN
T C: RETURN
490  VTAB (22): PRINT  TAB( 3);
"TEMPO";R: PRINT  TAB( 1);"JOGA
DA ";I
500  VTAB (19): PRINT "RESULTAD
O = ";
510  FOR D = 1 TO 5: VTAB (19):
 HTAB (9 + 4 * D): PRINT DC$(T(
D));: NEXT D
520  HTAB (16): VTAB (21): PRIN
T "ESCOLHA O GRUPO";
530 A = 1
540  HTAB (13): VTAB (3 + A): P
RINT  CHR$ (60);
550  GET B$: IF B$ <  > CHR$ (
81) AND B$ <  > CHR$ (90) AND
B$ <  > CHR$ (32) THEN 550
560  IF B$ =  CHR$ (32) THEN 62
0
570  HTAB (13): VTAB (3 + A): P
RINT " ";
580  IF B$ =  CHR$ (81) AND A >
 1 THEN A = A - 1
590  IF B$ =  CHR$ (90) AND A <
 12 THEN A = A + 1
600  PRINT  CHR$ (7)
610  GOTO 540
620  HTAB (13): VTAB (3 + A): P
RINT " ";: IF P(N,A) <  > 0 THE
N 950
630  IF A > 6 THEN 700
640 C = 0
650  FOR D = 1 TO 5: IF (T(D) =
 A) THEN C = C + 1
660  NEXT D
670 O(N,A) = C * A
680 P(N,A) = 1
690  RETURN
700  IF A = 11 THEN  FOR D = 1
TO 5:O(N,11) = O(N,11) + T(D):
NEXT :P(N,11) = 1: RETURN
710  FOR D = 1 TO 5:D(D) = 0: N
EXT :B = 0: FOR E = 1 TO 6:C =
0: FOR D = 1 TO 5: IF T(D) = E
THEN C = C + 1
720  NEXT D: IF C <  > 0 THEN B
 = B + 1
730  NEXT E
740 G = 1: FOR F = 1 TO 6: GOSU
B 1140: IF C <  > 0 THEN D(G) =
 F:G = G + 1
750  NEXT F
760 P(N,A) = 1:A = A - 6: ON A
GOTO 770,810,830,870,960,890
770  IF B > 2 THEN 900: IF B =
1 GOSUB 1160:O(N,7) = C * 4: RE
TURN
780 F = 1
790  GOSUB 1140:F = F + 1: IF C
 <  > 4 AND F <  > 7 THEN 790
795  IF C < 4 THEN 900
800 O(N,7) = 4 * (F - 1): RETUR
N
810  IF B <  > 2 THEN 900
812 F = D(1): GOSUB 1140: IF C
= 3 THEN 820
814 F = D(2): GOSUB 1140: IF C
<  > 3 THEN 900
820  FOR D = 1 TO 5:O(N,8) = O(
N,8) + T(D): NEXT : RETURN
```

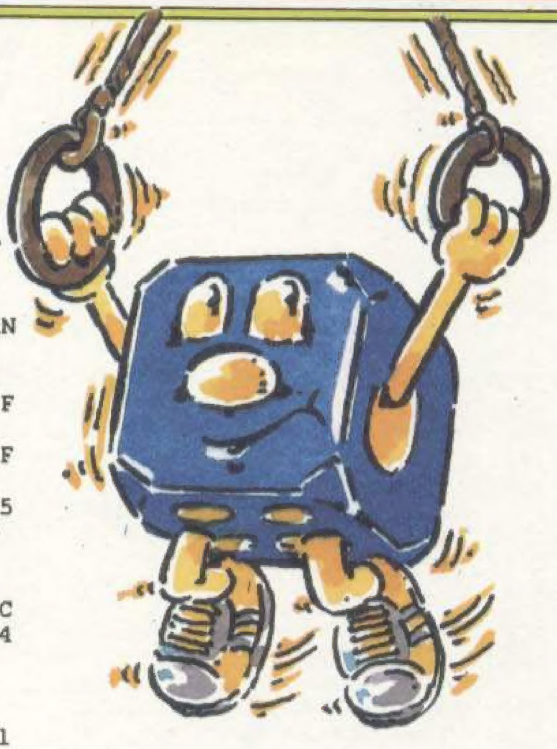
```
830  IF B <  > 4 THEN 850
835  GOSUB 1160: IF C <  > 18 A
ND C <  > 10 AND C <  > 14 OR (
C = 14 AND D(4) = 6) THEN 900
840 O(N,9) = 15: RETURN
850  IF B <  > 5 THEN 900
855  GOSUB 1160: IF C <  > 20 A
ND C <  > 15 AND C <  > 16 AND
C <  > 19 THEN 900
860  GOTO 840
870  IF B <  > 5 THEN 900
875  GOSUB 1160: IF C <  > 20 A
ND C <  > 15 THEN 900
880 O(N,10) = 30: RETURN
890  IF B <  > 1 THEN 900
895 O(N,12) = 50: PRINT  CHR$ (
7): FOR E = 1 TO 700: NEXT : RE
TURN
900  PRINT  CHR$ (7); CHR$ (7):
 HTAB (22): VTAB (5): PRINT "il
egal. ELIMINA?";
910  GET A$: IF A$ <  > "S" AND
 A$ <  > "N" THEN 910
920  HTAB (22): VTAB (5): PRINT
 CL$;
930  IF A$ = "N" THEN P(N,A + 6
) = 0: GOTO 530
940 P(N,A + 6) = 1: RETURN
950  PRINT  CHR$ (7); CHR$ (7):
 HTAB (22): VTAB (5): PRINT "GR
UPO OCUPADO";: FOR E = 1 TO 700
: NEXT
955  HTAB (22): VTAB (5): PRINT
 CL$;: GOTO 530
960  RETURN
1140 C = 0: FOR D = 1 TO 5: IF
T(D) = F THEN C = C + 1
1150  NEXT D: RETURN
1160 C = 0: FOR D = 1 TO B:C =
C + D(D): NEXT D: RETURN
```

5

```
430 BORDER 0: PAPER 0: INK 6:
CLS
440 PLOT 4,4: DRAW 0,167: DRAW
124,0: DRAW 0,-167: DRAW -124,
0
450 PRINT  INK 5;AT 1,5;N$(N);
INK 4;AT 2,1;"** PLACAR **"
460 RESTORE 1280: FOR M=4 TO
17: READ A$: PRINT AT M,1;A$;
Q$( TO 11-LEN A$);: IF M<>16
THEN  PRINT ":"
470 NEXT M
480 GOSUB 530
490 GOSUB 560: GOSUB 530
500 PRINT  FLASH 1;AT 20,18;"Q
UALQUER TECLA";AT 21,18;"P/ CO
NTINUAR  "
510 LET A$=INKEY$: IF A$=""
THEN  GOTO 510
520 RETURN
530 FOR D=1 TO 12: IF P(N,D)=1
THEN  PRINT AT 3+D,13;"X"
540 IF O(N,D)<>0 THEN  PRINT
AT 3+D,13;O(N,D)
550 NEXT D: LET C=0: FOR D=1
TO 12: LET C=C+O(N,D): NEXT D:
PRINT AT 17,13;C: RETURN
560 PRINT AT 8,18;"TEMPO ";R;
AT 9,18;"SECAO ";I
570 PRINT AT 2,18;"RESULTADO=
": LET T=-1: GOSUB 1180
580 PRINT AT 5,18;"ESCOLHA O G
RUPO"
590 LET A=4
600 PRINT AT A,15;CHR$ 150
610 LET B$=INKEY$: IF B$=""
THEN  GOTO 610
620 IF B$=" " THEN  LET A=A-3:
GOTO 710
630 IF B$="K" THEN  GOTO 650
640 IF B$<>"M" THEN  GOTO 610
650 PRINT AT A,15;" "
660 IF B$="K" AND A=4 THEN
GOTO 600
670 IF B$="M" AND A=15 THEN
GOTO 600
680 IF B$="M" THEN  LET A=A+1
690 IF B$="K" THEN  LET A=A-1
700 SOUND .01,5: GOTO 600
710 PRINT AT A+3,15;" ": IF P(
N,A)<>0 THEN  GOTO 1240
720 IF A>6 THEN  GOTO 780
730 LET C=0
740 FOR D=1 TO 5: IF T(D)=A
THEN  LET C=C+1
750 NEXT D
760 LET O(N,A)=C*A
770 LET P(N,A)=1: RETURN
780 IF A=11 THEN  FOR D=1 TO 5
: LET O(N,11)=O(N,11)+T(D):
NEXT D: LET P(N,11)=1: RETURN
790 FOR D=1 TO 5: LET D(D)=0:
NEXT D: LET B=0: FOR E=1 TO 6:
LET C=0: FOR D=1 TO 5: IF T(D)
=E THEN  LET C=C+1
800 NEXT D: IF C<>0 THEN  LET
B=B+1
810 NEXT E
820 LET G=1: FOR F=1 TO 6:
GOSUB 1250: IF C<>0 THEN  LET
D(G)=F: LET G=G+1
830 NEXT F
840 LET P(N,A)=1: IF A=7 THEN
GOTO 950
850 IF A=8 THEN  GOTO 1010
```

```
 860 IF A=9 THEN  GOTO 1050
 870 IF A=10 THEN  GOTO 1120
 890 IF A=12 THEN  GOTO 1160
 950 IF B>2 THEN  GOTO 1190
 960 IF B=1 THEN  GOSUB 1270:
 LET O(N,7)=C: RETURN
 970 LET F=1
 980 GOSUB 1250: LET F=F+1: IF
 C<>4 AND F<>7 THEN  GOTO 980
 990 IF C<4 THEN  GOTO 1190
1000 LET O(N,7)=4*(F-1): RETURN

1010 IF B<>2 THEN  GOTO 1190
1020 LET F=D(1): GOSUB 1250: IF
 C=3 THEN  GOTO 1040
1030 LET F=D(2): GOSUB 1250: IF
 C<>3 THEN  GOTO 1190
1040 LET O(N,8)=0: FOR G=1 TO 5
: LET O(N,8)=O(N,8)+T(G): NEXT
G: RETURN
1050 IF B<>4 THEN  GOTO 1080
1060 GOSUB 1270: IF C<>18 AND C
<>10 AND C<>14 OR (C=14 AND D(4
)=6) THEN  GOTO 1190
1070 LET O(N,9)=15: RETURN
1080 IF B<>5 THEN  GOTO 1190
1090 GOSUB 1270: IF C=15 OR C=1
6 OR C=19 THEN  GOTO 1070
1100 IF C<>20 THEN  GOTO 1190
1110 GOTO 1070
1120 IF B<>5 THEN  GOTO 1190
1130 GOSUB 1270: IF C=15 OR C=2
0 THEN  GOTO 1150
1140 GOTO 1190
1150 LET O(N,10)=30: RETURN
1160 IF B<>1 THEN  GOTO 1190
1170 LET O(N,12)=50: RETURN
1190 SOUND .5,5: PRINT AT 20,18
;"ILEGAL !";AT 21,18;"ELIMINA ?
"
1200 LET A$=INKEY$: IF A$="" TH
EN  GOTO 1200
1210 IF A$="N" THEN  PRINT AT 2
0,18;"          ";AT 21,18;"
      ": LET P(N,A)=0: GOTO 590
1220 IF A$<>"S" THEN  GOTO 1200
1230 LET P(N,A)=1: RETURN
1240 SOUND .5,5: PRINT AT 20,18
;"GRUPO OCUPADO": FOR H=1 TO 30
0: NEXT H: PRINT AT 20,18;"
          ": GOTO 590
1250 LET C=0: FOR D=1 TO 5: IF
T(D)=F THEN  LET C=C+1
1260 NEXT D: RETURN
1270 LET C=0: FOR D=1 TO B: LET
 C=C+D(D): NEXT D: RETURN
1280 DATA "UNS","DOIS","TRES","
QUATROS","CINCOS","SEIS","IGUAI
S","CASA CHEIA","CURTO","LONGO"
,"MISTO","YATCH","          ",
"TOTAL"
```



```
350 CLS
360 W=6:Y=0:GOSUB 980:PRINT"***
*SCORE****"
370 PRINT"UNS.........:"X$"DOIS
.........:"X$"TRES.........:"X$"Q
UATROS.....:"X$"CINCOS......:"X
$"SEIS........:"
380 PRINT"4 OF A KIND.:"X$"FULL
 HOUESE..;"X$"SHORT RUN...:"X$"
LONG RUN....:"X$"SHORT RUN...:"
X$"LONG RUN....:"X$"CHOICE.....
.:"X$"YATCH........:"X$
390 PRINT"TOTAL........:";
400 GOSUB 460
410 GOSUB 490:GOSUB 460
420 PRINT @466,"qualquer tecla"
;:PRINT @498,"para continuar";
430 SOUND 60,1
440 A$=INKEY$:IF A$="" THEN 440
450 RETURN
460 FOR D=1 TO 12:IF P(N,D)=1 T
HEN PRINT @45+D*32,"X";
470 IF O(N,D)<>0 THEN PRINT @45
+D*32,O(N,D);
480 NEXT D:C=0:FOR D=1 TO 12:C=
C+O(N,D):NEXT:PRINT @493,C;:RET
URN
490 PRINT @53,"ROUND"R;:PRINT @
84,"SECAO"I;
500 PRINT @115,"GRUPO FINAL=";
510 FOR D=1 TO 5:FOR G=1 TO 4:P
RINT @111+G*32-(D>3)*118+4*D,D$
(T(D),G);:NEXT G,D
520 PRINT @403,"SELECT GROUP";
530 A=1
540 PRINT @49+32*A,CHR$(95);
550 B$=INKEY$:IF B$<>" " AND B$
<>"^" AND B$<>CHR$(10) THEN 550
560 IF B$=" " THEN 620
570 PRINT @49+32*A," ";
580 IF B$="^" AND A>1 THEN A=A-
1
590 IF B$=CHR$(10) AND A<12 THE
N A=A+1
600 SOUND 200,1
610 GOTO 540
620 PRINT @49+32*A," ";:IF P(N,
A)<>0 THEN 950
630 IF A>6 THEN 700
640 C=0
650 FOR D=1 TO 5:IF T(D)=A THEN
 C=C+1
660 NEXTD
670 O(N,A)=C*A
680 P(N,A)=1
690 RETURN
700 IF A=11 THEN FOR D=1 TO 5:O
(N,11)=O(N,11)+T(D):NEXT:P(N,11
)=1:RETURN
710 FOR D=1 TO 5:D(D)=0:NEXT:B=
0:FOR E=1 TO 6:C=0:FOR D=1 TO 5
:IF T(D)=E THEN C=C+1
720 NEXT D:IF C<>0 THEN B=B+1
730 NEXT E
740 G=1:FOR F=1 TO 6:GOSUB 1140
:IF C<>0 THEN D(G)=F:G=G+1
750 NEXT F
760 P(N,A)=1:A=A-6:ON A GOTO 77
0,810,830,870,960,890
770 IF B>2 THEN 900 ELSE IF B=1
 GOSUB 1160:O(N,7)=C*4:RETURN
780 F=1
790 GOSUB 1140:F=F+1:IF C<>4 AN
D F<>7 THEN 790 ELSE IF C<4 THE
N 900
800 O(N,7)=4*(F-1):RETURN
810 IF B<>2 THEN 900 ELSE F=D(1
):GOSUB 1140:IF C=3 THEN 820 EL
SE F=D(2):GOSUB 1140:IF C<>3 TH
EN 900
820 FOR D=1 TO 5:O(N,8)=O(N,8)+
T(D):NEXT:RETURN
830 IF B<>4 THEN 850 ELSE GOSUB
 1160:IF C<>18 AND C<>10 AND C<
>14 OR (C=14 AND D(4)=6) THEN 9
00
840 O(N,9)=15:RETURN
850 IF B<>5 THEN 900 ELSE GOSUB
 1160:IF C<>20 AND C<>15 AND C<
>16 AND C<>19 THEN 900
860 GOTO 840
870 IF B<>5 THEN 900 ELSE GOSUB
 1160:IF C<>20 AND C<>15 THEN 9
00
880 O(N,10)=30:RETURN
890 IF B<>1 THEN 900 ELSE O(N,1
2)=50:SOUND 5,8:FOR E=1 TO 700:
NEXT E:RETURN
900 SOUND 20,1:PRINT @432,"ileg
al. PERDE?";
910 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 910
920 PRINT @432,"
";
930 IF A$="N" THEN P(N,A+6)=0:G
OTO 530
940 P(N,A+6)=1:RETURN
950 SOUND 5,1:PRINT @433,"SECAO
 PREENCHIDA";:FOR E=1 TO 700:NE
XT:PRINT @433,"
";:GOTO 530
960 RETURN
1140 C=0:FOR D=1 TO 5:IF T(D)=F
 THEN C=C+1
1150 NEXT D:RETURN
1160 C=0:FOR D=1 TO B:C=C+D(D):
NEXT D:RETURN
```

IMPRESSÃO DO ESQUELETO

A primeira seção dessa rotina — que vai até a linha 520 no Spectrum e linha 450 nos outros micros — mostra na te-

la o esqueleto do placar (nomes dos grupos e do jogador) e chama duas outras rotinas para preencher os grupos e receber a última entrada.

A rotina constituída pelas três linhas seguintes preenche o esqueleto com o placar anterior e fornece a soma dos totais. Depois dela encontra-se uma rotina maior, incumbida de receber e analisar as entradas.

As primeiras linhas dessa rotina apresentam na tela informações sobre o número da partida e da jogada e, o que é mais importante, mostram a combinação final obtida. As linhas seguintes, além de identificar a tecla que está sendo pressionada, movem o cursor para cima e para baixo no esqueleto. Para a movimentação do cursor, o Spectrum usa as teclas M e K; o MSX e o TRS-Color, as setas, e o Apple e o TK-2000, as teclas Q e Z.

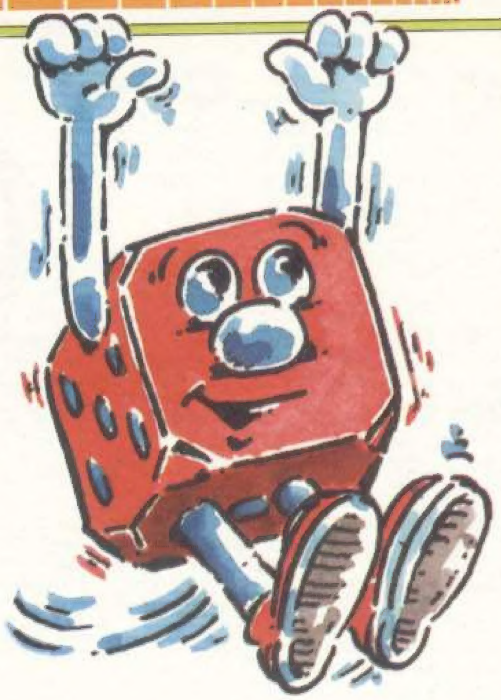
Quando se pressiona a barra de espaço, o programa interpreta que o grupo apontado pelo cursor foi escolhido. A posição do cursor é armazenada na variável **A**, onde **A** = 1 corresponde à posição do grupo dos UM, **A** = 2 equivale à posição do grupo dos DOIS e assim por diante até **A** = 11, que corresponde à posição do YATCH. O valor de **A** é usado pelo restante da rotina para calcular a pontuação do placar.

Uma outra variável importante é a variável **B**, que contém o número de dados diferentes entre si: **B** = 2 (dois tipos de valores), por exemplo, representaria uma FULL HOUSE e **B** = 5 (cinco tipos de valores), um LONGO.

Portanto, dependendo do valor de **A**, rotinas diferentes são chamadas para verificar se a combinação dos dados realmente corresponde ao grupo escolhido. Se corresponder, o placar é calculado e apresentado na tela. Caso contrário, a mensagem "ILEGAL" é exibida. O jogador decide, então — com um S ou um N —, se o grupo deve ser eliminado. O computador verifica também quais grupos já foram usados.

RESULTADO FINAL

Esta última seção simplesmente apresenta na tela o placar final, com os pontos dos jogadores em cada partida. O número da partida é exibido no topo do placar final e os nomes dos jogadores mais abaixo, à esquerda.



```
990 CLS:PRINT">>>>>>>>>>     PLACAR FINAL     <<<<<<<<<<"
1000 LOCATE0,4:FORD=1TONP:PRINT N$(D):NEXT
1020 FORD=1TONP
1030 C=0:FORE=1TO12:C=C+O(D,E):NEXT:S(D,R)=C:NEXTD
1040 FORD=1TO5:LOCATE3+D*4,2:PRINTD;:NEXT
1050 PRINT" TOTAL"
1060 FORD=1TONP
1070 FORE=1TOR
1080 LOCATE3+E*4,3+D:PRINTS(D,E);:NEXTE,D
1090 FORD=1TONP:C=0:FORE=1TOR:C=C+S(D,E):NEXTE:LOCATE26,3+D:PRINTC;:NEXTD
1100 LOCATE2,19:PRINT"< qualquer tecla para outra rodada >"
1110 A$=INKEY$:IFA$=""THEN1110
1120 FORE=1TONP:FORD=1TO12:O(E,D)=0:P(E,D)=0:NEXTD,E
1130 RETURN
```

```
990  HOME : INVERSE : PRINT ">>>>>>>>>>  PLACAR FINAL  <<<<<<<<<<": NORMAL
1000  VTAB (5): FOR D = 1 TO NP: PRINT  LEFT$ (N$(D),10): NEXT

1020  FOR D = 1 TO NP
1030 C = 0: FOR E = 1 TO 12:C = C + O(D,E): NEXT :S(D,R) = C: NEXT D
1040  FOR D = 1 TO 5: HTAB (9 + D * 4): VTAB (3): PRINT D;: NEXT
1050  PRINT "   TOTAL"
1060  FOR D = 1 TO NP
1070  FOR E = 1 TO R
1080  HTAB (9 + E * 4): VTAB (4 + D): PRINT S(D,E);: NEXT E,D
1090  FOR D = 1 TO NP:C = 0: FOR E = 1 TO R:C = C + S(D,E): NEXT E: HTAB (33): VTAB (4 + D): PRINT C;: NEXT D
1100  HTAB (2): VTAB (22): PRINT "< QUALQUER TECLA PARA OUTRA RODADA >";
1110  GET A$: IF A$ = "" THEN 1110
1120  FOR E = 1 TO NP: FOR D = 1 TO 12:O(E,D) = 0:P(E,D) = 0: NEXT D,E
1130  RETURN
```

```
1290 BORDER 7: PAPER 7: INK 3: CLS
1300 PRINT "**********PLACAR FINAL**********"
1310 PRINT AT 7,0: FOR D=1 TO NP: PRINT N$(D);" :"'': NEXT D
1320 PRINT  INK 1;AT 3,0;"JOGADOR"; INK 2;AT 3,13;"T E M P O"
1330 PRINT AT 5,12;: FOR D=1 TO 5: PRINT  INK 2;TAB (4+D*4);D;: NEXT D: PRINT  INK 0;"   TOTE"
1340 FOR D=1 TO NP
1360 LET C=0: FOR E=1 TO 12: LET C=C+O(D,E): NEXT E: LET S(D,R)=C: NEXT D
1370 FOR D=1 TO NP: FOR E=1 TO R
1380 PRINT  INK 4;AT 6+D*2,4+E*4;S(D,E): NEXT E: NEXT D
1390 FOR D=1 TO NP: LET C=0: FOR E=1 TO R: LET C=C+S(D,E): NEXT E: PRINT AT 6+D*2,28;C: NEXT D
1400 PRINT #1; INVERSE 1; INK 0;" QUALQUER TECLA PARA CONTINUAR "
1410 LET A$=INKEY$: IF A$="" THEN  GOTO 1410
1420 FOR E=1 TO NP: FOR D=1 TO 12: LET O(E,D)=0: LET P(E,D)=0: NEXT D: NEXT E
1430 RETURN
```

```
990 CLS:PRINT"*****TABELA DE PONTOS*****"
1000 PRINT @128:FOR D=1 TO NP:PRINT N$(D):NEXT
1010 PRINT @75,"R O U N D"
1020 FOR D=1 TO NP
1030 C=0:FOR E=1 TO 12:C=C+O(D,E):NEXT:S(D,R)=C:NEXT D
1040 FOR D=1 TO 5:PRINT @98+D*4,D;:NEXT
1050 PRINT" TOTAL"
1060 FOR D=1 TO NP
1070 FOR E=1 TO R
1080 PRINT @129+D*32+E*4,S(D,E);:NEXT E,D
1090 FOR D=1 TO NP:C=0:FOR E=1 TO R:C=C+S(D,E):NEXT E:PRINT @153+D*32,C;:NEXT D
1100 PRINT @448,"  QUALQUER TECLA PARA PROXIMA      JOGADA.";
1110 A$=INKEY$:IF A$="" THEN 1110
1120 FOR E=1 TO NP:FOR D=1 TO 12:O(E,D)=0:P(E,D)=0:NEXT D,E
1130 RETURN
```

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

CÓDIGO DE MÁQUINA

Atice as cobras: a vida de Willie vai ficar bem complicada, mas *Avalanche* ganhará muita emoção.

PROGRAMAÇÃO BASIC

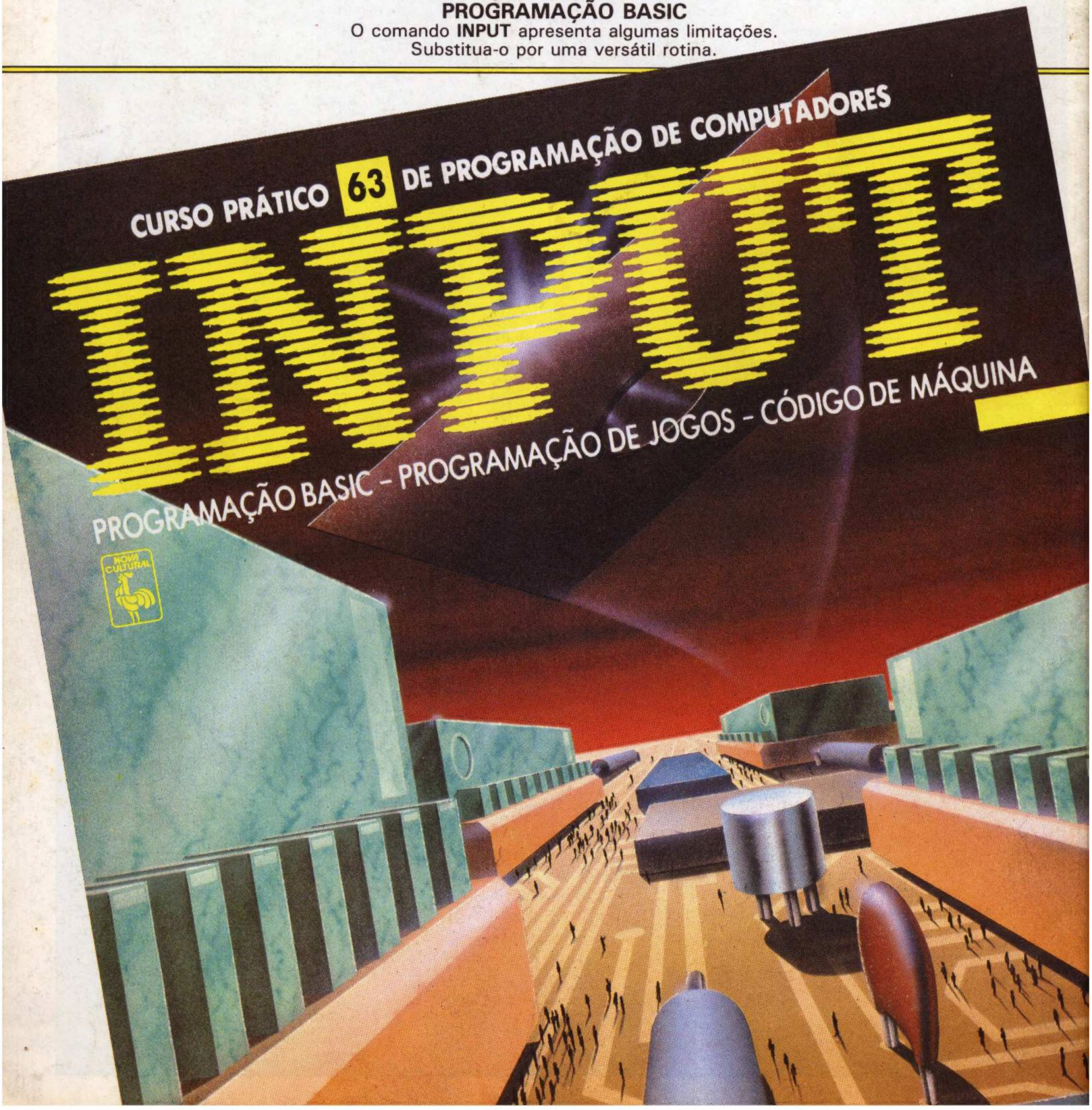
Sem o sistema operacional, seria muito difícil trabalhar com o computador.
Conheça esse importante componente do seu micro.

PROGRAMAÇÃO BASIC

Aprenda a lidar com arquivos. A manipulação dessa ferramenta é fundamental para determinadas operações.

APLICAÇÕES

Você quer comprar um monitor? Avalie a qualidade de sua imagem com nosso programa para teste de vídeo.

PROGRAMAÇÃO BASIC

O comando **INPUT** apresenta algumas limitações.
Substitua-o por uma versátil rotina.